L Number	Hits	Search Text	DB	Time stamp
1	1	arbcritical	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/08/14 16:21
-	629	(710/113-118).CCLS.	USPAT	2003/08/13 16:42
-	250	(710/306).CCLS.	USPAT	2003/08/13 16:43
-	107	(710/309).CCLS.	USPAT	2003/08/13 16:43
-	367	(710/240).CCLS.	USPAT	2003/08/13 16:43
-	58	(710/243).CCLS.	USPAT	2003/08/13 17:14
-	19	critical adj request	USPAT	2003/08/13 16:51
-	91363	input/output	USPAT	2003/08/13 16:52
-	32	"memory control hub"	USPAT	2003/08/13 16:52
-	672	(710/305).CCLS.	USPAT	2003/08/13 17:15
-	4	"memory control hub" and ((710/305).CCLS.)	USPAT	2003/08/13 17:31
-	165801	arbit$	USPAT	2003/08/13 17:32
-	275	arbit$ and ((710/305).CCLS.)	USPAT	2003/08/14 10:27
-	71	(710/242).CCLS.	USPAT	2003/08/14 10:32
-	33	(710/121).CCLS.	USPAT	2003/08/14 10:32
-	392655	isochronous critical	USPAT	2003/08/14 10:32
-	8	((710/121).CCLS.) and (isochronous critical)	USPAT	2003/08/14 10:35
-	14	((710/242).CCLS.) and (isochronous critical)	USPAT	2003/08/14 10:40
-	4	(request same priority same critical) and ((710/242).CCLS.)	USPAT	2003/08/14 10:42
-	237	request same priority same critical	USPAT	2003/08/14 10:45
-	33	"memory control hub"	USPAT	2003/08/14 10:46
-	0	"io control hub"	USPAT	2003/08/14 10:46
-	2375	(input output) same control same hub	USPAT	2003/08/14 10:47
-	25	"memory control hub" and ((input output) same control same hub)	USPAT	2003/08/14 10:51
-	1	("memory control hub" and ((input output) same control same hub)) and "62"	USPAT	2003/08/14 10:51
-	0	("memory control hub" and ((input output) same control same hub)) and ((710/242).CCLS.)	USPAT	2003/08/14 11:10
-	88	critical same arbit$ same priority$	USPAT	2003/08/14 11:11
-	88	critical same arbit$ same priority	USPAT	2003/08/14 11:11
-	92	critical same arbit$ same priorit$	USPAT	2003/08/14 11:14
-	1921	(710/107-124).CCLS.	USPAT	2003/08/14 11:15
-	672	(710/305).CCLS.	USPAT	2003/08/14 11:20
-	888	(710/305-306,710/309).CCLS.	USPAT	2003/08/14 14:03

-	120	(710/242-243,710/240).CCLS.	USPAT	2003/08/14 11:20
-	8	(critical same arbit$ same priorit$) and ((710/242-243,710/240).CCLS.)	USPAT	2003/08/14 11:20
-	2	(critical same arbit$ same priorit$) and ((710/305-306,710/309).CCLS.)	USPAT	2003/08/14 11:21
-	0	(critical same arbit$ same priorit$) and ((710/305).CCLS.)	USPAT	2003/08/14 11:21
-	33	(critical same arbit$ same priorit$) and ((710/107-124).CCLS.)	USPAT	2003/08/14 11:21
-	36	((critical same arbit$ same priorit$) and ((710/242-243,710/240).CCLS.)) ((critical same arbit$ same priorit$) and ((710/305-306,710/309).CCLS.)) ((critical same arbit$ same priorit$) and ((710/107-124).CCLS.))	USPAT	2003/08/14 14:00
-	213	critical same request same line	USPAT	2003/08/14 14:18
-	14320	("710").CLAS.	USPAT	2003/08/14 14:20
-	39	(critical same request same line) and (("710").CLAS.)	USPAT	2003/08/14 15:08
-	39	((critical same arbit$ same priorit$) and ((710/242-243,710/240).CCLS.)) ((critical same arbit$ same priorit$) and ((710/305-306,710/309).CCLS.)) ((critical same arbit$ same priorit$) and ((710/107-124).CCLS.))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/08/14 15:10

US006393505B1

(12) **United States Patent**
Scalise et al.

(10) **Patent No.: US 6,393,505 B1**
(45) **Date of Patent: May 21, 2002**

(54) **METHODS AND APPARATUS FOR DATA BUS ARBITRATION**

(75) Inventors: **Albert M. Scalise**, San Jose; **Jano D. Banks**, Cupertino, both of CA (US)

(73) Assignee: **DVDO, Inc.**, Sunnyvale, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/227,502**

(22) Filed: **Jan. 6, 1999**

(51) **Int. Cl.**[7] **G06F 13/00**

(52) **U.S. Cl.** **710/107**; 710/40; 710/113; 710/240; 710/241; 710/244

(58) **Field of Search** 710/113, 36, 40, 710/107, 111, 240, 241, 244

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,557,608	A *	9/1996	Calvignac et al.	370/389
5,905,998	A *	5/1999	Ebrahim et al.	711/144
5,956,341	A *	9/1999	Galand et al.	370/412
6,041,039	A *	3/2000	Kilkki et al.	370/230

OTHER PUBLICATIONS

Micron Technology Inc., *Technical Note, Achieve Maximum Compatibility In SDRAM/SGRAM Design*, Compatibility in SDRAM/SGRAM Design, May, 1997.

Micron Technology Inc., *Synchronous DRAM*, 16 MEG: x16 SDRAM, Oct., 1997.

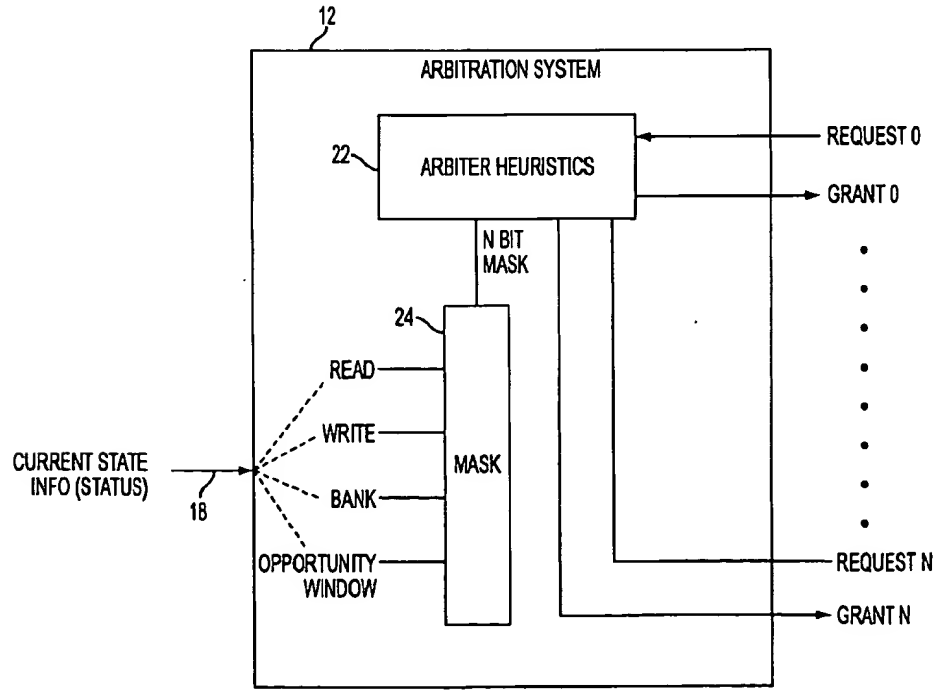
* cited by examiner

Primary Examiner—Ario Etienne
(74) *Attorney, Agent, or Firm*—Oppenheimer Wolff & Donnelly, LLP

(57) **ABSTRACT**

A data bus arbitration system is disclosed including a bus status monitor which is coupled to a data bus and generates a bus status signal for use by an arbiter. The arbiter is coupled to a number of requesters, each of which belongs to a distinct class of requesters. The arbiter arbitrates between multiple requests using heuristics dependent upon the classes of the requesters. The nature of one class of requestors is that the requestors have real time requirements which must be met in order to maintain data integrity within the system. The nature of a second class of requestors is such that the requestors have semi-real time requirements which must be met in order to maintain data integrity within the system. The nature of the system is such that the available bandwidth must be utilized very efficiently in order to maintain data integrity within the system. The arbiter system disclosed grants access to the requesters using the heuristics disclosed while maintaining an efficiency of at least 80% of the total bandwidth for all requestors.

20 Claims, 10 Drawing Sheets



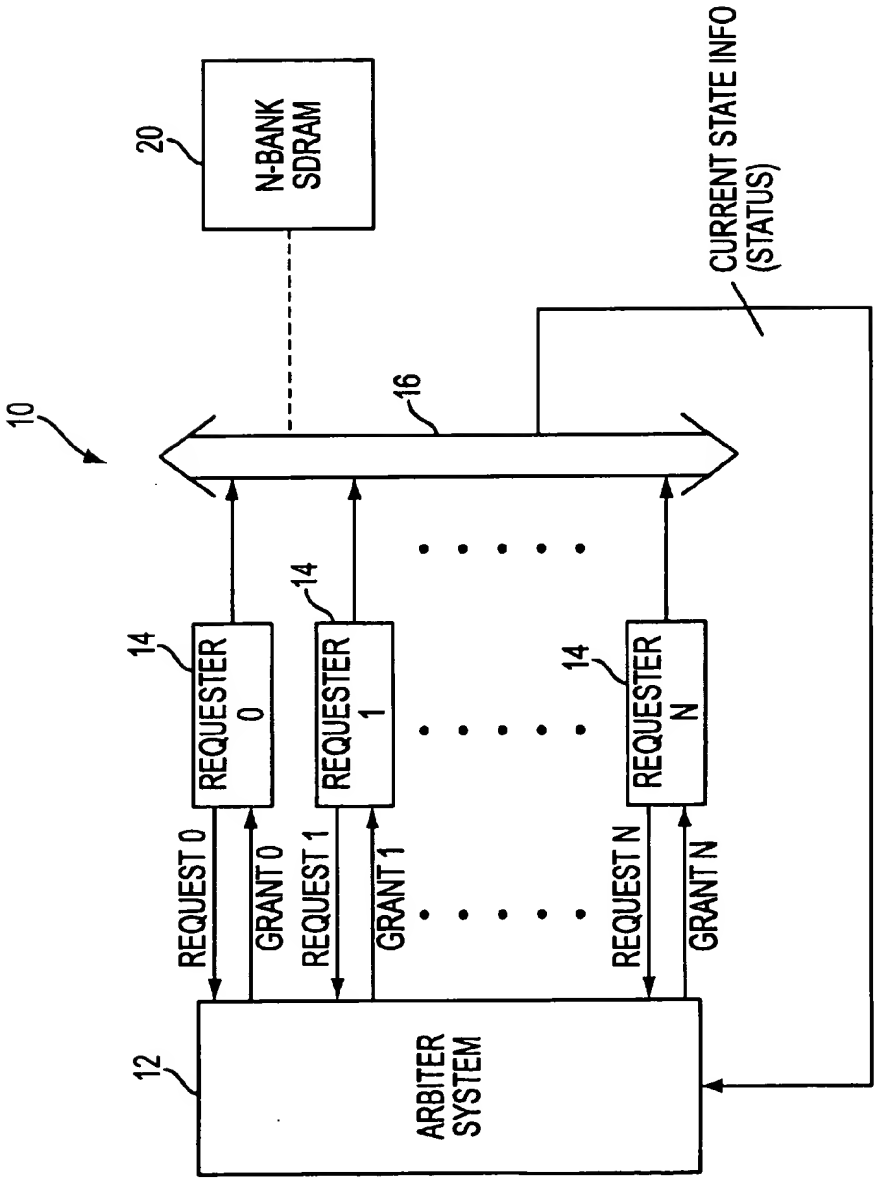

20
N-BANK SDRAM
CURRENT STATE INFO (STATUS)
10
16
14
REQUESTER 0
14
REQUESTER 1
14
REQUESTER N
REQUEST 0
GRANT 0
REQUEST 1
GRANT 1
REQUEST N
GRANT N
12
ARBITER SYSTEM

FIG. 1

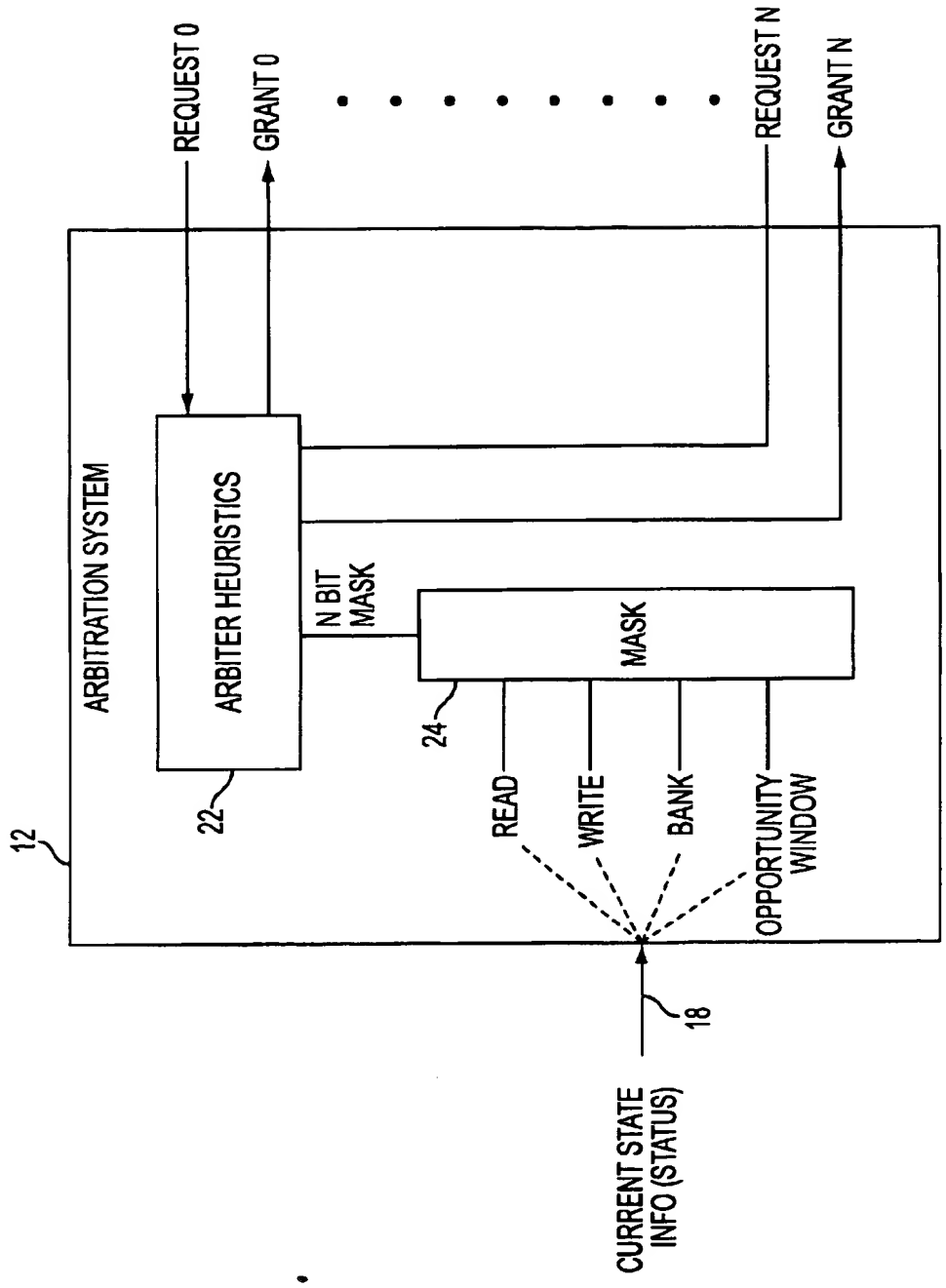

REQUEST 0
GRANT 0
REQUEST N
GRANT N
ARBITRATION SYSTEM
ARBITER HEURISTICS
N BIT MASK
MASK
22
24
12
READ
WRITE
BANK
OPPORTUNITY WINDOW
18
CURRENT STATE INFO (STATUS)

FIG. 2

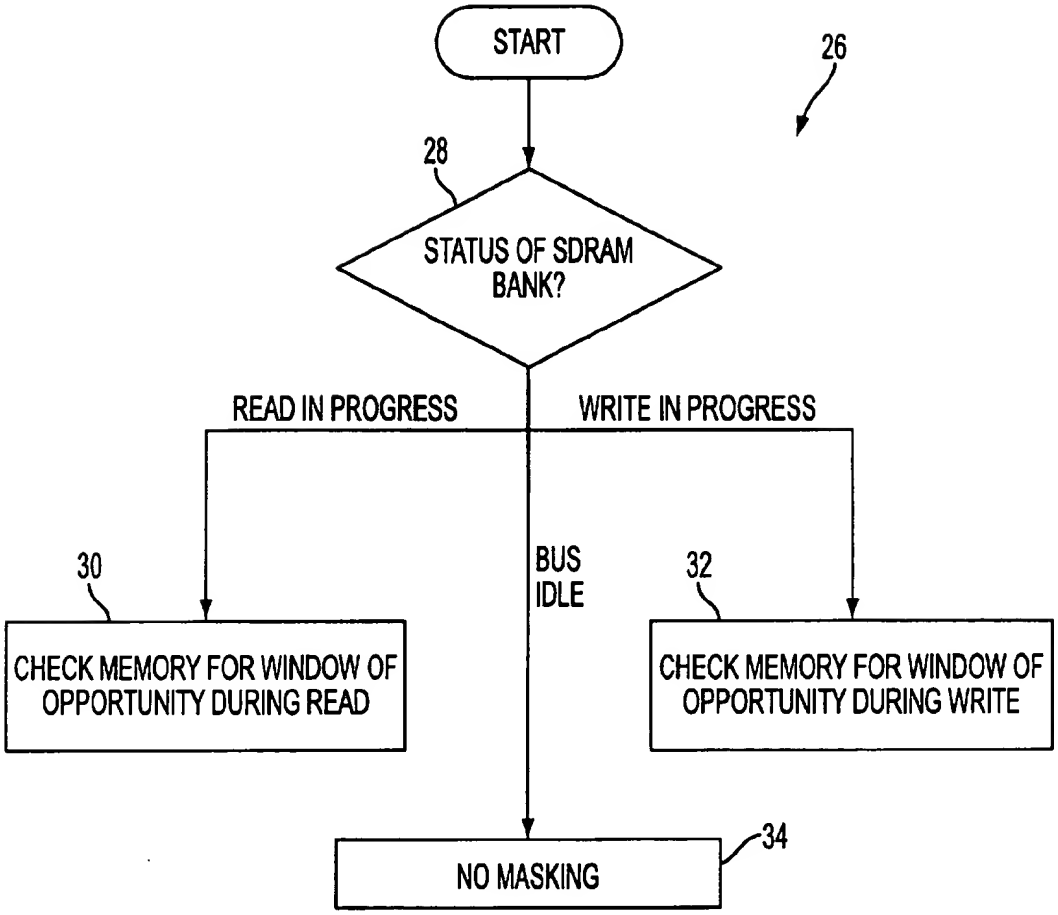

START
26
28
STATUS OF SDRAM BANK?
READ IN PROGRESS
WRITE IN PROGRESS
BUS IDLE
30
32
CHECK MEMORY FOR WINDOW OF OPPORTUNITY DURING READ
CHECK MEMORY FOR WINDOW OF OPPORTUNITY DURING WRITE
34
NO MASKING

FIG. 3

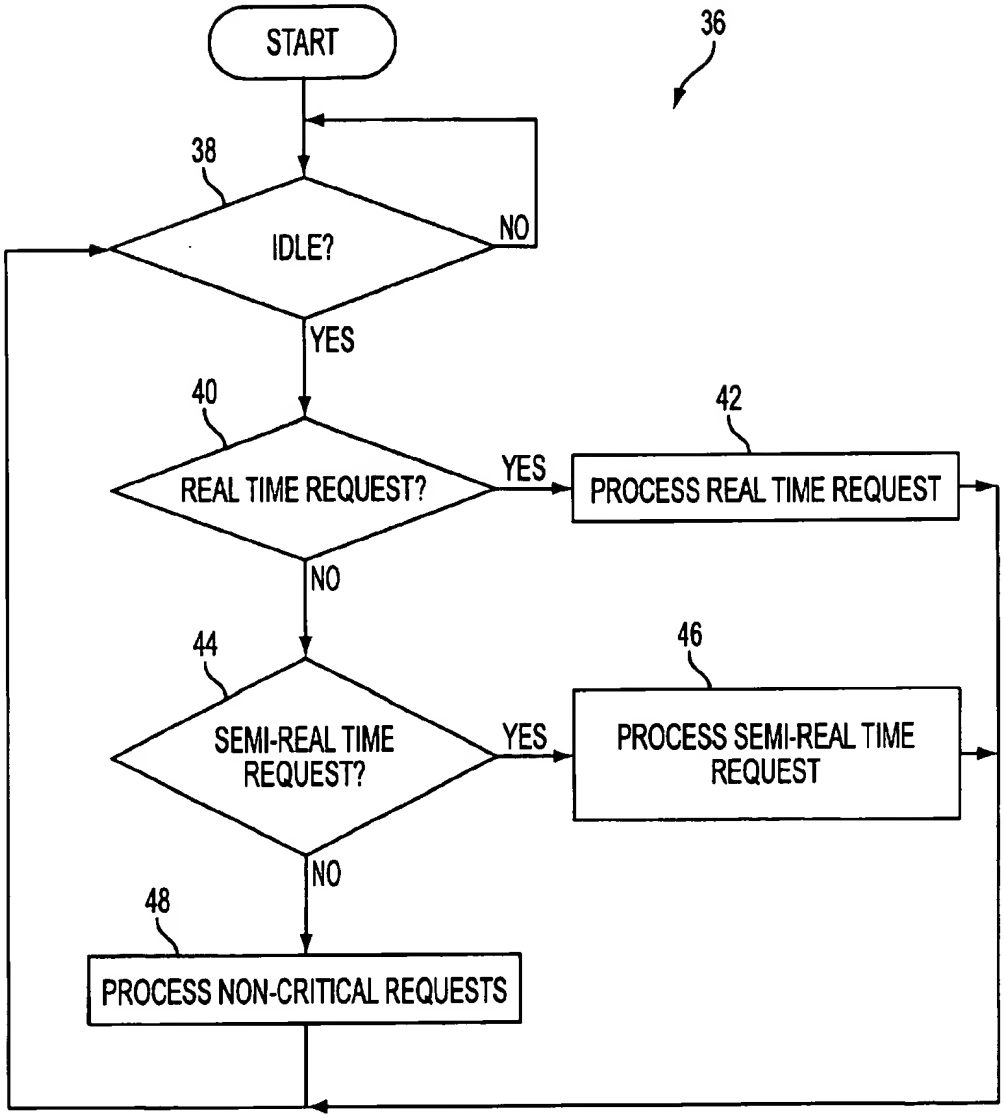

START
36
38
IDLE?
NO
YES
40
REAL TIME REQUEST?
YES
42
PROCESS REAL TIME REQUEST
NO
44
SEMI-REAL TIME REQUEST?
YES
46
PROCESS SEMI-REAL TIME REQUEST
NO
48
PROCESS NON-CRITICAL REQUESTS

FIG. 4

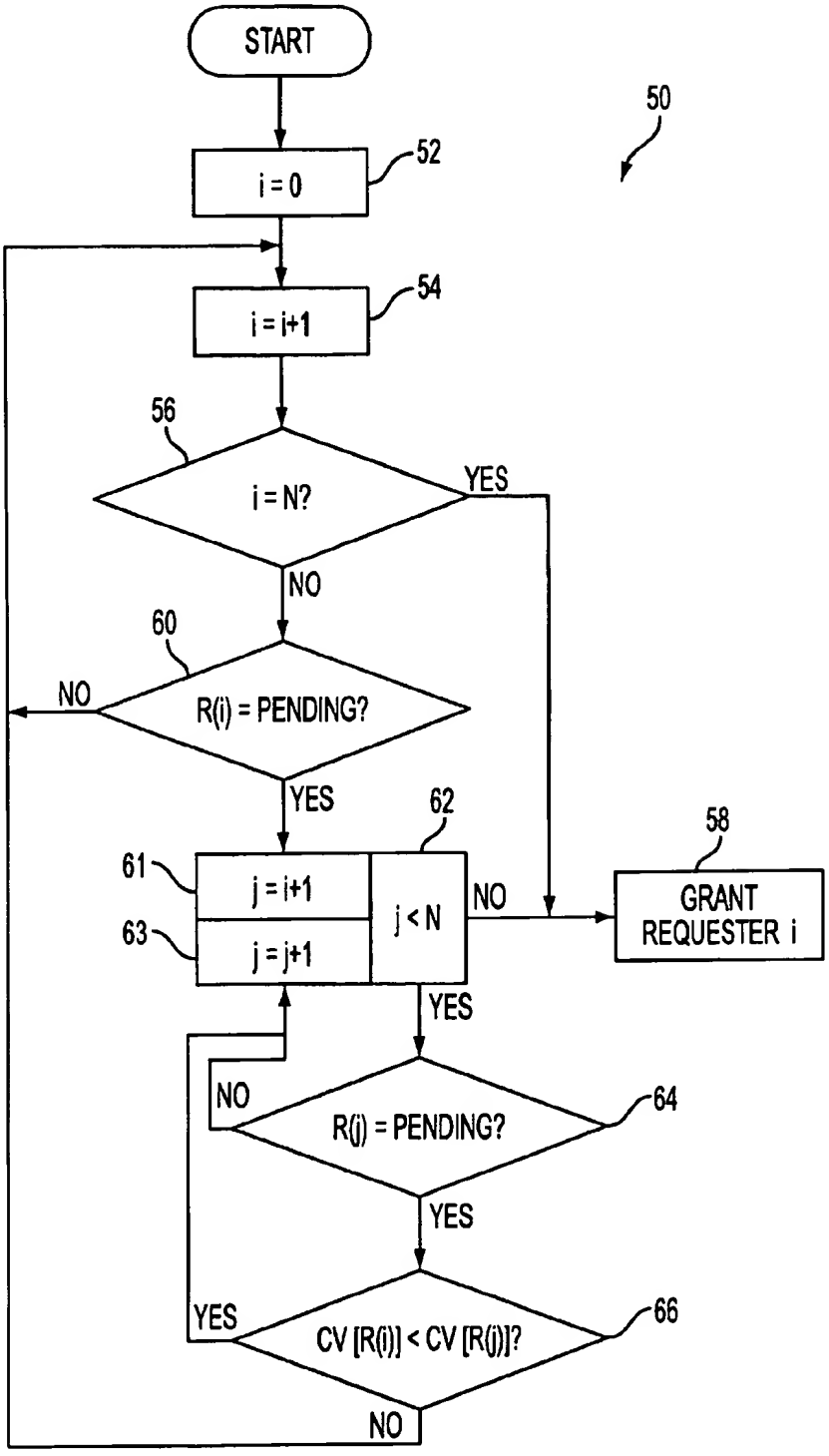

START
50
52
i = 0
54
i = i+1
56
i = N?
YES
NO
60
R(i) = PENDING?
NO
YES
62
58
61
j = i+1
63
j = j+1
j < N
NO
GRANT REQUESTER i
YES
NO
64
R(j) = PENDING?
YES
YES
66
CV [R(i)] < CV [R(j)]?
NO

FIG. 5

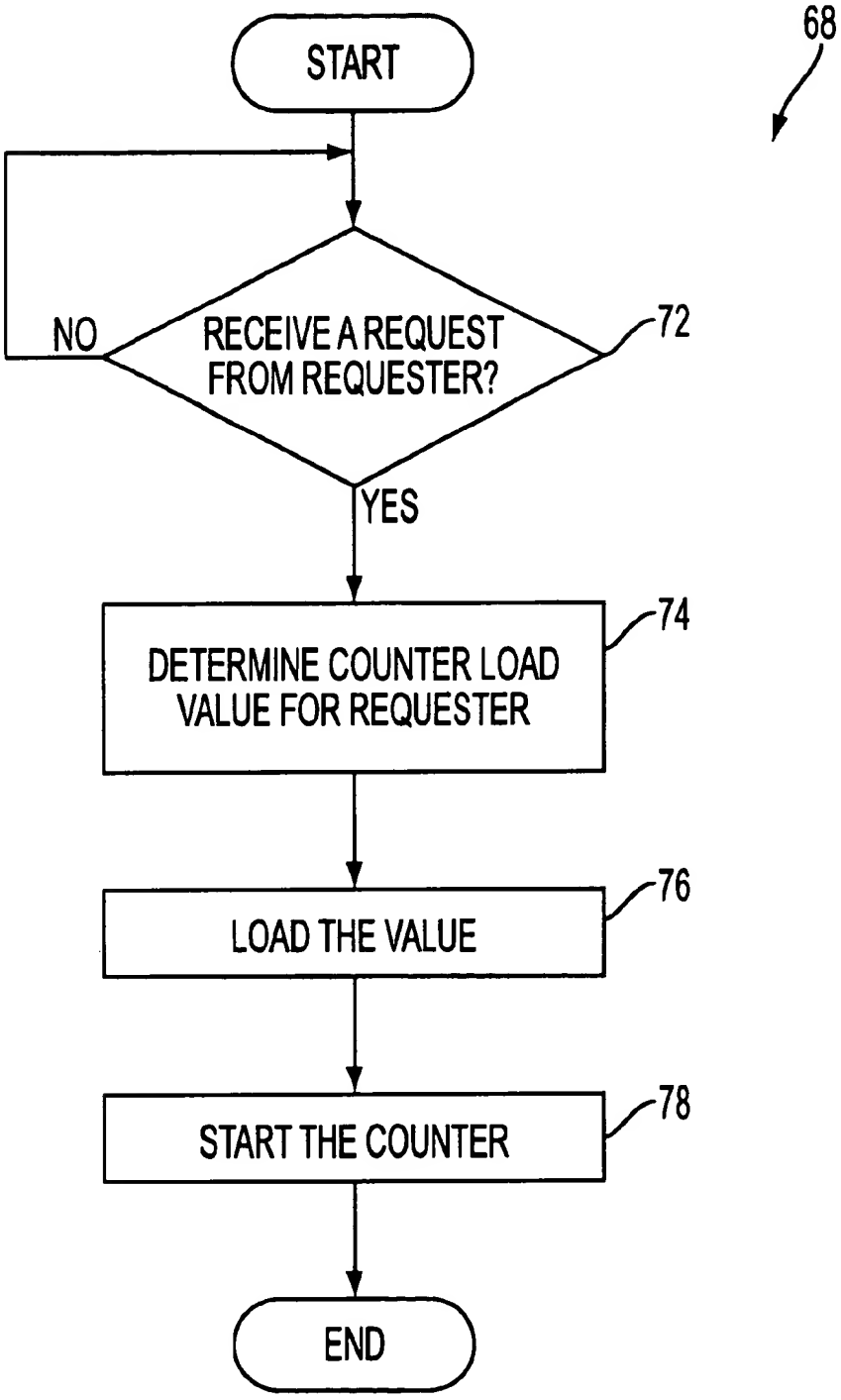

68
START
RECEIVE A REQUEST FROM REQUESTER?
72
NO
YES
DETERMINE COUNTER LOAD VALUE FOR REQUESTER
74
LOAD THE VALUE
76
START THE COUNTER
78
END

FIG. 6

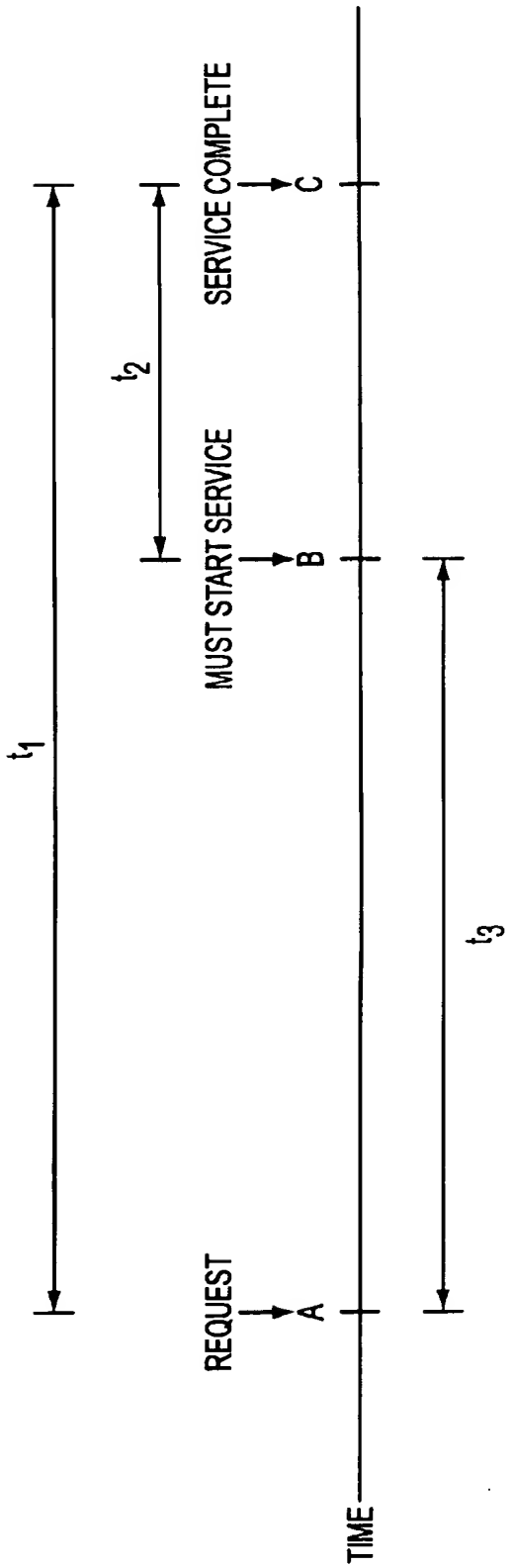

SERVICE COMPLETE
C
MUST START SERVICE
B
REQUEST
A
TIME
t1
t2
t3

FIG. 7

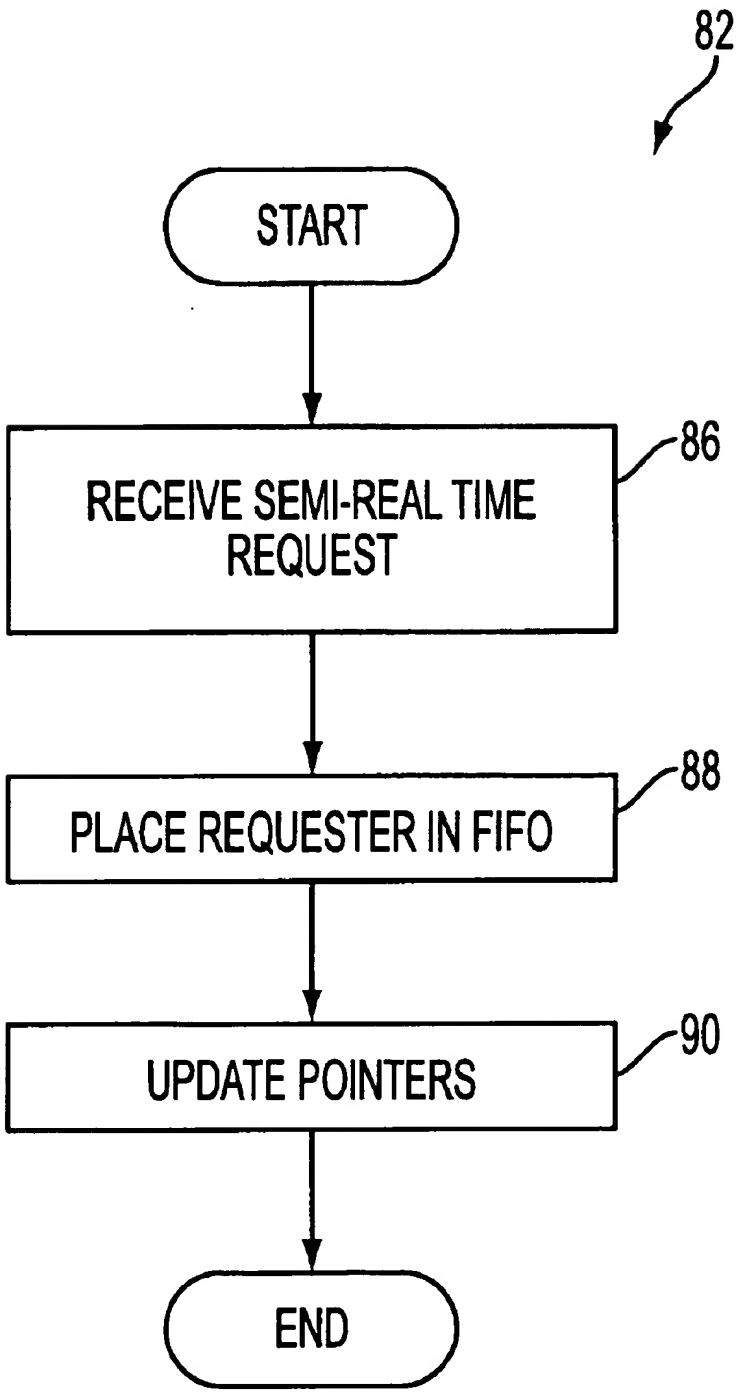

82
START
RECEIVE SEMI-REAL TIME REQUEST
86
PLACE REQUESTER IN FIFO
88
UPDATE POINTERS
90
END

FIG. 8

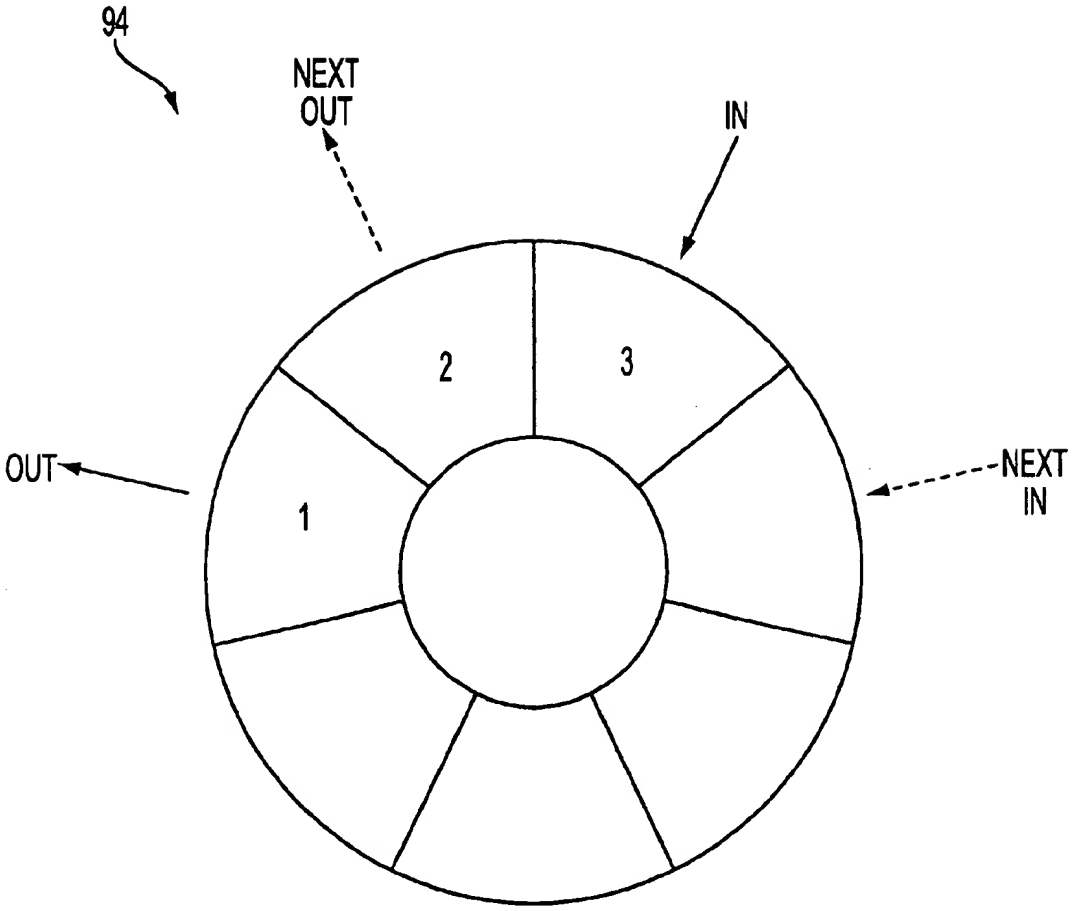


FIG. 9

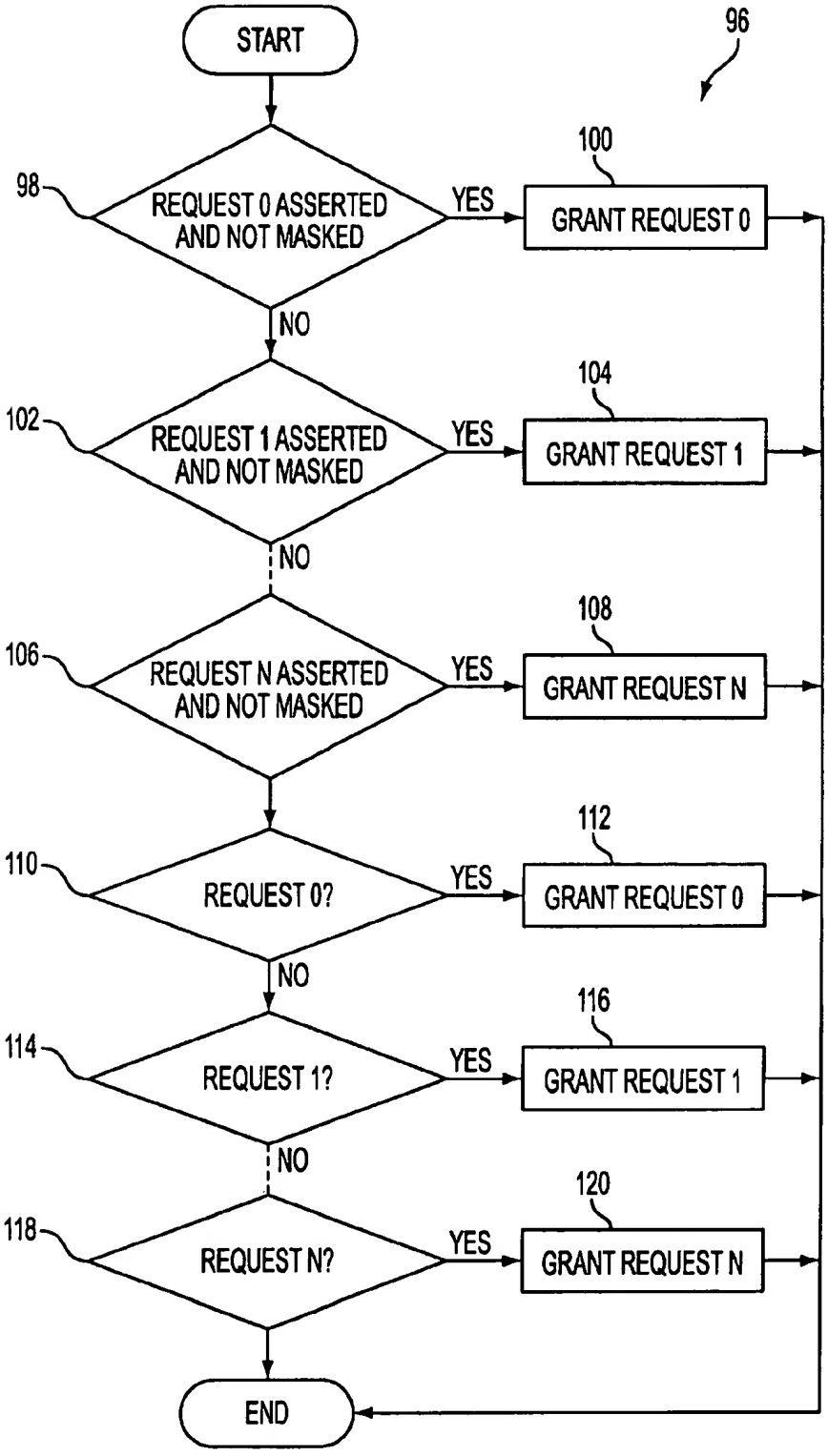

96
START
98
REQUEST 0 ASSERTED AND NOT MASKED
YES
100
GRANT REQUEST 0
NO
102
REQUEST 1 ASSERTED AND NOT MASKED
YES
104
GRANT REQUEST 1
NO
106
REQUEST N ASSERTED AND NOT MASKED
YES
108
GRANT REQUEST N
110
REQUEST 0?
YES
112
GRANT REQUEST 0
NO
114
REQUEST 1?
YES
116
GRANT REQUEST 1
NO
118
REQUEST N?
YES
120
GRANT REQUEST N
END

FIG. 10

METHODS AND APPARATUS FOR DATA BUS ARBITRATION

CROSS REFERENCE TO RELATED APPLICATIONS

This application is related to co-pending U.S. patent application Ser. No. 09/226,776 entitled Methods And Apparatus For Variable Length SDRAM Transfers filed on the same day, which is incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to arbitration for access to a digital device and more particularly, to techniques for arbitrating multiple data transfer requests to optimize the operation of a computer system.

2. Description of the Related Art

Dynamic random access memory (DRAM) is used to provide a number of different functions in computers including: "scratch pad" memory and video frame buffers. Synchronous dynamic random access memory (SDRAM) is a type of DRAM designed to deliver bursts of data at very high speed using automatic addressing, multiple page interleaving, and a synchronous (or clocked) interface. The SDRAM is typically monitored and managed by a SDRAM controller.

The SDRAM controller receives data transfer requests for memory access to the SDRAM. To establish the priority of the data transfer requestors, the SDRAM controller generally includes an arbiter. The arbiter receives information from the requestors to determine the type of data transfer requested, and then uses an arbitration scheme to assign a priority to each of the requesters. The SDRAM controller then grants memory access to the SDRAM based on the priority established by the arbiter.

Perhaps the simplest type of arbitration scheme is fixed priority, where each type of request in a system is assigned a fixed priority. When an arbiter receives multiple requests, the request with the highest priority always receives access to the data bus. For example, if there is a request A of high priority and request B of medium priority, then request B will receive memory access only if request A is not present.

A major problem with fixed priority arbitration schemes is that they can handle only a limited number of requesters because of the risk that devices with a low priority will not be timely serviced, if at all. For example, in a fixed priority scheme, if there is a request A of high priority, request B of medium priority, and request C of low priority, and the arbiter is constantly bombarded with requests A and B, then request C will "starve" and never be serviced. The risk of starvation rises with the number of devices that request service. In fact, if a fixed priority scheme includes ten devices with ten levels of priority, starvation is virtually guaranteed.

Another type of arbitration scheme used in the prior art is a round robin scheme. The standard round robin scheme deals with all the requests in a fixed order. For example, if request A is of high priority, request B of medium priority, and request C of low priority, the round robin scheme will grant request A first, then grant request B, and then grant request C. However, if request A is not present, then request B becomes of high priority, request C of medium priority and request A of low priority. Therefore, the round robin scheme continually shifts priority between all of the devices it services.

As with the fixed priority scheme, round robin schemes also run into starvation problems when handling a large number of requesters. Take the example of a round robin scheme that looks for and grants requests A–Z in alphabetical order, where request A has not been asserted at the time it is sought. If request A is asserted at the time that request B is being sought, then it is possible that request A will not be granted until requests B–Z have been serviced.

Another type of arbitration scheme used in the prior art is a dynamic priority scheme, which utilizes a set of rules that determine the priority of the requestors established by the arbiter. For example, the rules may give a higher priority to requesters that require the most bandwidth of memory access from a SDRAM or give higher priority to requestors that have most recently been serviced.

Dynamic priority schemes allow for in theory, an unlimited number of ways for an arbiter to establish the priority of multiple requestors. However, the more complicated the rules are, the more process overhead will be required to maintain the dynamic priority scheme. This is particularly true in cases where the dynamic priority scheme must arbitrate between real time devices.

There are several other arbitration schemes in the prior art such as a window based scheme that gives each requestors a certain slice of time where memory access will be granted. The advantage of the window based scheme is that it can guarantee bandwidth for its requesters. However, as with the dynamic priority scheme, the windows based scheme involving multiple requestors would be very complicated and require a great deal of logic. Furthermore, a window based scheme would have difficulty maintaining efficient data bus utilization because a slice in time may be set aside for a device when there is no request resulting in idle tine where the bus does nothing.

In view of the foregoing, it is desirable to have methods and an apparatus that is able to establish priority between multiple devices with real time requirements, while at the same time maintaining efficient utilization of the data bus.

SUMMARY OF THE INVENTION

The present invention fills these needs by providing methods and an apparatus for arbitrating between and servicing requestors having real time requirements while maintaining efficient utilization of the data bus. It should be appreciated that the present invention can be implemented in numerous ways, including as a process, an apparatus, a system, a device or a method. Several inventive embodiments of the present invention are described below.

In one embodiment of the present invention, a data bus arbitration system is disclosed. The data bus arbitration system includes a bus status monitor which is coupled to a data bus and generates a bus status signal for use by an arbiter. The arbiter is coupled to number of requesters, each of which belong to a distinct class of requesters. The arbiter arbitrates between multiple requests using heuristics dependent upon the classes of the requesters. The arbiter grants access to one of the requestors to the data bus while maintaining an efficiency of at least 80% of a total data bandwidth of the data bus for all requesters. One of the distinct classes of requesters is preferably a real time class. The data bus arbitration system preferably includes a timer that provides an indication of when access to the data bus must be granted for a real time request.

In another embodiment of the present invention, a method for data bus arbitration is disclosed. The method includes monitoring a status of a data bus. A data bus request from a

requestor belonging to a first class of requesters is first processed by a first heuristic method. The method then proceeds with second processing a data bus request from a requestor belonging to a second class of requestors by a second heuristic method. Access to the data bus is granted to a requestor based upon the status of the data bus, the first processing and the second processing, understanding that the first class of requestors is of a generally higher priority than the second class of requestors. The method preferably includes third processing a data bus request from a requester belonging to a third class of requesters by a third heuristic method, understanding that the second class of requesters is of a generally higher priority than the third class of requestors.

In yet another embodiment of the present invention, a method for data bus arbitration is disclosed. The method includes means for monitoring a status of a data bus. A data bus request from a requester belonging to a first class of requesters is processed by a first processing means using a first heuristic method. The method then proceeds with processing a data bus request using a second processing means from a requester belonging to a second class of requesters by a second heuristic method. Access to the data bus is granted to a requestor based upon the status of the data bus, the first processing means and the second processing means, understanding that the first class of requestors is of a generally higher priority than the second class of requestors.

An advantage of the present invention is that it provides for arbitrating between and granting requestors having real time requirements while preventing starvation of all devices in need of service. The arbiter of the present invention is able to grant priority based not only on the type of request received, but also based on the requirements of the request. Furthermore, the present invention maintains efficient utilization of the data bus.

Other aspects and advantages of the invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrating by way of example the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be readily understood by the following detailed description in conjunction with the accompanying drawings. To facilitate this description, like reference numerals designate like structural elements.

FIG. **1** is a block diagram of a SDRAM system in accordance with one embodiment of the present invention.

FIG. **2** illustrates arbiter system in accordance with one embodiment of the present invention.

FIG. **3** is a flow chart of a method of generating masks from mask generator.

FIG. **4** is a flow chart illustrating a method of processing requests for memory access by arbiter heuristics module.

FIG. **5** is a flow chart of a method of the operation to process a real time request in arbiter heuristics module to optimize utilization of the SDRAM bus.

FIG. **6** is a flow chart of a method of operating a counter in arbiter heuristics module for each real time requestor on the system.

FIG. **7** is a time line representing operation for generating a counter load value.

FIG. **8** is a flow chart of a method of the operation of processing a semi-real time request in arbiter heuristics module to optimize the SDRAM bus.

FIG. **9** is a diagram of an N deep FIFO used by method to prioritize the real time requests.

FIG. **10** is a flow chart of a method of operation for processing non-critical requests in arbiter heuristics module to optimize the SDRAM bus.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be understood, however, to one skilled in the art, that the present invention may be practiced without some or all of these specific details. In other instances, well known process operations have not been described in detail in order not to unnecessarily obscure the present invention.

FIG. **1** is a block diagram of a SDRAM system **10** in accordance with one embodiment of the present invention. SDRAM system **10** includes an arbiter system **12**, a number of requesters **14**, a data bus **16**, and an N-Bank SDRAM **20**, which stores and retrieves data for an infinite number of banks for use by an SDRAM controller and the devices to which the SDRAM controller is attached. Arbiter system **12** receives requests for memory access to N-Bank SDRAM **20** from requesters **14**. After establishing the priority of the requests received, arbiter system **12** grants memory access to requestors **14** based on the current state of N-Bank SDRAM **20** communicated from data bus **16**. Requestors **14** then access N-Bank SDRAM through data bus **16**.

FIG. **2** illustrates arbiter system **12** in accordance with one embodiment of the present invention. Arbiter system **12** includes an arbiter heuristics module **22** and a mask generator **24**. Based on the current state information regarding N-Bank SDRAM **20**, mask generator **24** creates a mask to optimize the use of the SDRAM, and communicates the information to arbiter heuristics module **22**. The current state information includes the identity of the bank, whether there is a read in progress, a write in progress, and whether there is a window of opportunity for overlapping transactions.

Arbiter heuristics module **22** receives requests from requesters **14** and uses a set of heuristic schemes in addition to mask information from mask generator **24** to establish the priority of the requests. Arbiter heuristics module **22** then grants memory access to requesters **14** based on priority and masking for optimization of the SDRAM bus. For more detail regarding the use of current state information to overlap data transactions, please see cross referenced U.S. patent application entitled Methods And Apparatus For Variable Length SDRAM Transfers.

FIG. **3** is a flow chart of a method **26** of generating masks from mask generator **24**. Method **26** begins with an operation **28** which determines whether the status of a SDRAM bank is a read in progress, a write in progress or bus idle. If there is a read in progress, then an operation **30** checks memory for a window of opportunity during the read. If there is a write in progress, then an operation **32** checks memory for a window of opportunity during the write. If operation **30** or operation **32** discovers a window of opportunity in which to overlap transactions, then a mask is generated and transmitted to arbiter heuristics module **22**. If operation **28** determines that the bus is idle, then an operation **34** provides that there be no masking of the request.

FIG. **4** is a flow chart illustrating a method **36** of processing requests for memory access by arbiter heuristics module **22**. Method **36** begins with an operation **38** which waits until the data bus is at idle. When the data bus is idle, an operation **40** determines whether the request for memory access is a real time request. If there is a real time request, it is assigned the highest priority and processed by an operation **42**, after which method **36** starts again at operation **38**. If the request is not a real time request, then an operation

44 determines whether the request is a semi-real time request. If there is a semi-real time request, then it is processed by an operation 46, after which method 36 starts again at operation 38. If the request is neither a real time request or a semi-real time request, then it is classified as a non-critical request and processed in an operation 48, after which method 36 repeats at operation 38.

FIG. 5 is a flow chart of a method 50 of operation 42 to process a real time request in arbiter heuristics module 22 to optimize utilization of the SDRAM bus. In method 50, the counter associated with each of the pending real time requests is compared to the counters for all other pending real time requests, to establish the priority of processing the real time requests. Method 50 determines the priority of a real time request based on the real time requirements of the real time requests.

Method 50 begins in an operation 52 where i is set equal to 0, and a counter is started by incrementing i by one (i=i+1) in an operation 54. An operation 56 determines whether i equals N, which is the total number of real time requesters in the system. If i does equal N, requester i is granted in an operation 58 because method 50 incremented i through all of the real time requesters. Therefore the last requestor where i=N must the highest priority. If i does not equal N, then method 50 proceeds to an operation 60 which determines whether or not requester i is pending. A requester is defined as pending if it is asserted and not masked. If requestor i is not pending, then i is incremented again in operation 54.

If requester i is pending, then a loop starts in an operation 61 to find out if requestor i is the request with the highest priority. This is accomplished by setting j equal to i+1 in operation 61. An operation 62 checks if j represents an existing requester by making sure j is less than or equal to N. If j is greater than N, requester i is the last requester, and of highest priority, therefore it is granted in operation 58. If j<N, then an operation 64 determines whether requestor j is pending. If it is not pending, an operation 63 increments the value of j by one, and then returns to operation 62.

If requester j is pending, then CV[R(i)], the counter value of requestor i is compared to CV[R(j)], the counter value of requestor j in an operation 66. If $CV[R(i)] \leqq CV[R(j)]$, then requestor i has a greater priority than requester j, and method 50 returns to operation 63. If CV[R(i)]>CV[R(j)], then requestor j has a greater priority than requestor i, and method 50 moves on to check another real time request by incrementing i again by returning to operation 54.

FIG. 6 is a flow chart of a method 68 of operating a counter in arbiter heuristics module 22 for each real time requestor on the system. All instances of method 68 run in parallel with method 50 described in FIG. 5. Method 68 begins by waiting until a request has been received from a requester in an operation 72. A counter load value is generated in an operation 74 based on the latency and bus time requirements of the requestor. The counter load value, which reflects the real time and bandwidth requirements of the request, is then loaded in an operation 76, and the counter is started in anoperation78.

FIG. 7 is a time line 80 representing operation 74 for generating a counter load value. Point A represents the time a request is received, point C represents the time at which service of the request must be completed, and point B represents the time service for the request must be started in order for the request to be completely serviced by point C. Because time t_1, the length of time in which a request can be serviced, and time t_2, the actual length of time a request requires to be serviced are known, time t_3 is determined and utilized by method 50 to determine the required priority of the real time requesters.

FIG. 8 is a flow chart of a method 82 of operation 46 of processing a semi-real time request in arbiter heuristics module 22 to optimize utilization of the SDRAM bus. Method 82 begins when a semi-real time request is received in an operation 86. An operation 88 places the requestor into a first in first out (FIFO) unit. Pointers are updated in an operation 90, and method 82 ends.

FIG. 9 is a diagram of an N deep FIFO 94 used by method 82 to prioritize the real time requests.

FIG. 10 is a flow chart of a method 96 of operation 48 for processing non-critical requests in arbiter heuristics module 22 to optimize utilization of the SDRAM bus. Method 96 begins with an operation 98, which determines whether or not request 0 is asserted and not masked. If request 0 is asserted and not masked, then an operation 100 grants request 0. If not, an operation 102 determines whether or not request 1 is asserted and not masked. If request 1 is asserted and not masked, then an operation 104 grants request 1. The process is repeated for all requests 2-N.

Although method 96 is for processing non-critical requests, by searching for requests that are asserted and not masked and granting them first, the resources of the SDRAM can be even further optimized. If no non-masked requests exist, then method 96 proceeds to an operation 110 which simply determines whether or not request 0 exists. If request 0 does exist, then an operation 112 grants request 0. The process is again repeated for all requests 1-N.

In summary, the present invention provides for arbitrating between and granting requesters having real time requirements while preventing starvation of all devices in need of service. Furthermore, the present invention maintains efficient utilization of the data bus. For example, if a real time request is received by the arbiter of the present invention, it will automatically be given a higher priority than all semi-real time requests and all non-critical requests.

The arbiter then compares the requirements of the real time requests with the requirements of all the other real time requests. The requirements include information regarding the time by which a real time request must be serviced, and also the length of time it will take to service the particular request. In this manner that the arbiter is able to determine priority based on which particular real time request has the most restrictive pending time requirement, and is therefore in most "urgent" need of service.

The invention has been described herein in terms of several preferred embodiments. Other embodiments of the invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention. Furthermore, certain terminology has been used for the purposes of descriptive clarity, and not to limit the present invention. The embodiments and preferred features described above should be considered exemplary, with the invention being defined by the appended claims.

What is claimed is:

1. A data bus arbitration system comprising:

a bus status monitor coupled to a data bus and developing a bus status signal including at least a window of opportunity signal; and

an arbiter coupled to plurality of requesters, wherein each of said plurality of requestors belongs to one of a plurality of classes of requestors, and wherein a total usage of said plurality of requesters of said data bus is at least 80% of a total data bandwidth of said data bus, said plurality of classes including a real-time class, said arbiter arbitrating between multiple requests using heuristics dependent upon said classes of said requestors and upon said bus status signal, and granting access to one of said requesters to said data bus, wherein said heuristics for said real time class include the latency and bus time requirements of said requestors belonging to said real time class.

2. A data bus arbitration system as recited in claim 1 wherein said total usage of said plurality of requestors of said data bus is at least 90% of said total bandwidth of said data bus.

3. A data bus arbitration system as recited in claim 1 wherein said latency and bus time requirements include a timer associated with a request from a real-time requester which aids in the arbitration between multiple requests.

4. A data bus arbitration system as recited in claim 3 wherein said timer provides an indication of when access to said data bus must be granted.

5. A data bus arbitration system as recited in claim 1 wherein a class of requestor includes a semi real-time requester, wherein requestors in said real-time class are of a generally higher priority than requesters in said semi real-time class.

6. A data bus arbitration system as recited in claim 5 wherein said heuristic includes a first-in-first-out (FIFO) heuristic.

7. A data bus arbitration system as recited in claim 5 wherein said classes of requesters includes a non-critical class.

8. A data bus arbitration system as recited in claim 7 wherein requesters in said semi real-time class are of a generally higher priority than requesters in said non-critical class.

9. A data bus arbitration system as recited in claim 7 wherein said heuristic for said non-critical class is a fixed priority heuristic.

10. A data bus arbitration system as recited in claim 1 wherein said bus status signal is a mask.

11. A method for data bus arbitration comprising:

monitoring a status of a data bus including at least a window of opportunity signal;

first processing a data bus request from a requestor belonging to a first class of requesters by a first heuristic method, wherein said first class of requestors includes a real-time class, and wherein said first heuristic method for said real time class includes looking at the latency and bus time requirements of said requesters belonging to said real time class;

second processing a data bus request from a requestor belonging to a second class of requesters by a second heuristic method, wherein said first class of requestors is of a generally higher priority than said second class of requestors; and

granting access to said data bus to a requestor based upon said status of said data bus, said first processing and said second processing.

12. A method for data bus arbitration as recited in claim 11 further comprising:

third processing a data bus request from a requestor belonging to a third class of requestors by a third heuristic method, wherein said second class of requestors is of a generally higher priority than said third class of requestors.

13. A method for data bus arbitration as recited in claim 11 wherein said first heuristic method includes a timer which determines when said requestor must be serviced, and wherein said second heuristic method includes a first-in-first-out (FIFO) heuristic.

14. A method for data bus arbitration as recited in claim 12 wherein said first heuristic method includes a timer which determines when said requestor must be serviced, said second heuristic method includes a first-in-first-out (FIFO) heuristic, and wherein said third heuristic is a fixed priority heuristic.

15. A data bus arbitration system comprising:

a monitor operative to monitor a status of a data bus including at least a window of opportunity signal; and

a processor operative to processes a data bus request from a requestor belonging to a real-time class of requestors by a heuristic method, wherein said heuristic method includes looking at the latency and bus time requirements of said requester;

wherein a requestor is granted access to said data bus based upon said status of said data bus and said heuristic method.

16. A data bus arbitration system as recited in claim 15 wherein said heuristic method includes a timer which determines when said requestor must be serviced.

17. A data bus arbitration system as recited in claim 16 wherein said timer indicates a priority time period in which a service for a request must be started in order to completely service the request.

18. A data bus arbitration system as recited in claim 17 wherein said priority time period is at least partially derived from a total service time less the time required to process the request, said total service time being the time period from a time the request is received to a time the request must be completely serviced.

19. A method for data bus arbitration comprising:

monitoring a status of a data bus including at least a window of opportunity signal;

first processing a data bus request from a requestor belonging to a first class of requestors by a first heuristic method, wherein said first class of requesters includes a real-time class, and wherein said first heuristic method for said real time class includes looking at the latency and bus time requirements of said requesters belonging to said real time class;

second processing a data bus request from a requestor belonging to a second class of requestors by a second heuristic method, wherein said second class of requesters includes a semi-real-time class, and wherein said first class of requesters is of a generally higher priority than said second class of requestors;

third processing a data bus request from a requestor belonging to a third class of requesters by a third heuristic method, wherein said third class of requesters includes a non-critical class, and wherein said first and second class of requesters are of a generally higher priority than said third class of requesters; and

granting access to said data bus to a requester based upon said status of said data bus, said first processing, said second processing and said third processing.

20. A data bus arbitration system comprising:

a bus status monitor coupled to a data bus and developing a bus status signal including at least a window of opportunity signal; and

an arbiter coupled to plurality of requestors, wherein each of said plurality of requesters belongs to one of a plurality of classes of requesters, and wherein a total usage of said plurality of requestors of said data bus is at least 80% of a total data bandwidth of said data bus, said plurality of classes including at least one of a real-time class, a semi-real time class and a non-critical class, said arbiter arbitrating between multiple requests using heuristics dependent upon said classes of said requestors and upon said bus status signal, and granting access to one of said requesters to said data bus, wherein said heuristics for said real time class include the latency and bus time requirements of said requesters belonging to said real time class, and wherein said heuristic for said semi-real time class includes a first-in-first-out (FIFO) heuristic, and wherein said heuristic for said non-critical class is a fixed priority heuristic.

* * * * *

US005872936A

United States Patent [19]

Eckstein

[11] Patent Number: 5,872,936

[45] Date of Patent: Feb. 16, 1999

[54] **APPARATUS FOR AND METHOD OF ARBITRATING BUS CONFLICTS**

[75] Inventor: **Bruce Eckstein**, Cupertino, Calif.

[73] Assignee: **Apple Computer, Inc.**, Cupertino, Calif.

[21] Appl. No.: **646,076**

[22] Filed: **May 7, 1996**

Related U.S. Application Data

[63] Continuation of Ser. No. 436,968, May 8, 1995, abandoned.

[51] **Int. Cl.**[6] **G06F 13/36**

[52] **U.S. Cl.** **395/287**; 345/520; 348/94

[58] **Field of Search** 395/287, 520, 395/521, 293, 299; 345/520, 521; 348/94

[56] **References Cited**

U.S. PATENT DOCUMENTS

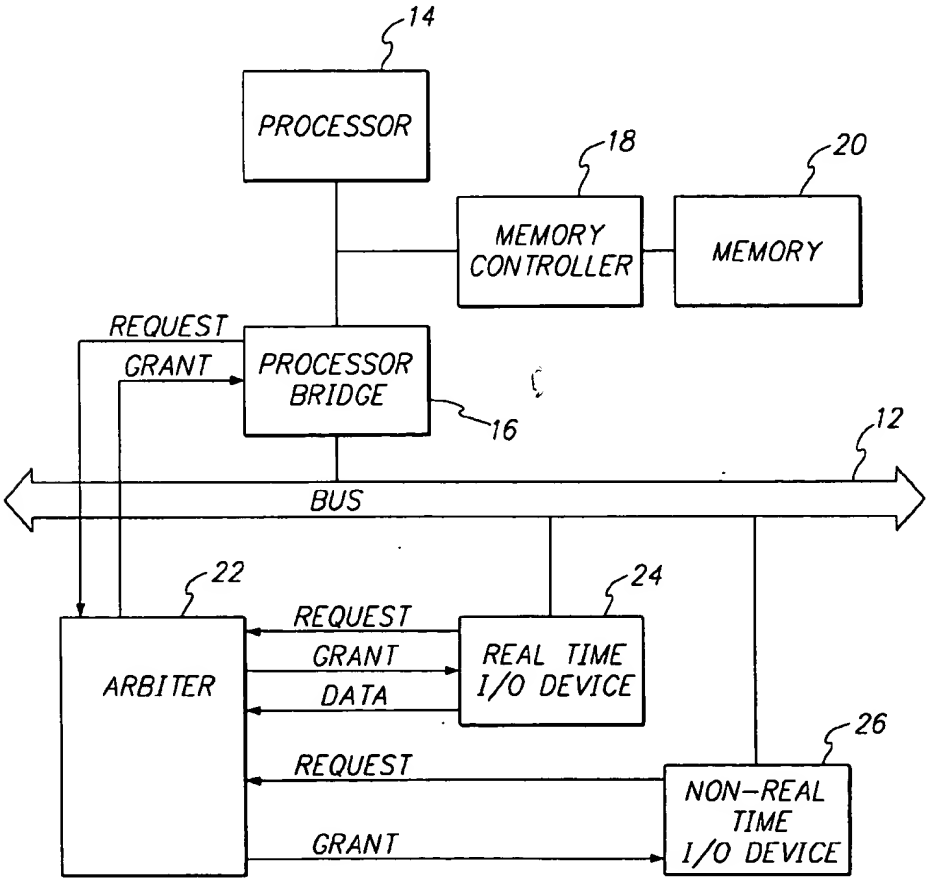
Re. 34,810	12/1994	Lemaine et al. .	
4,701,796	10/1987	Kamiya .	
4,723,164	2/1988	Nienaber .	
4,899,218	2/1990	Waldecker et al. .	
4,991,112	2/1991	Callemyn	395/520
5,001,652	3/1991	Thompson	395/520
5,197,119	3/1993	Ohuchi	395/520
5,323,309	6/1994	Taylor et al. .	
5,392,396	2/1995	MacInnis .	
5,467,295	11/1995	Young et al. .	

Primary Examiner—Meng-Ai T. An
Assistant Examiner—X. Chung-Trans
Attorney, Agent, or Firm—Burns, Doane, Swecker & Mathis, L.L.P.

[57] **ABSTRACT**

An apparatus for and method of arbitrating bus conflicts is disclosed. The system can use information about the status of a data stream to switch between more or less restrictive prioritization schemes. In one particular embodiment, the arbitration system can also be programmed by causing it to store information on overall system requirements for data which the arbitration system can then use to set a current prioritization scheme.

26 Claims, 8 Drawing Sheets



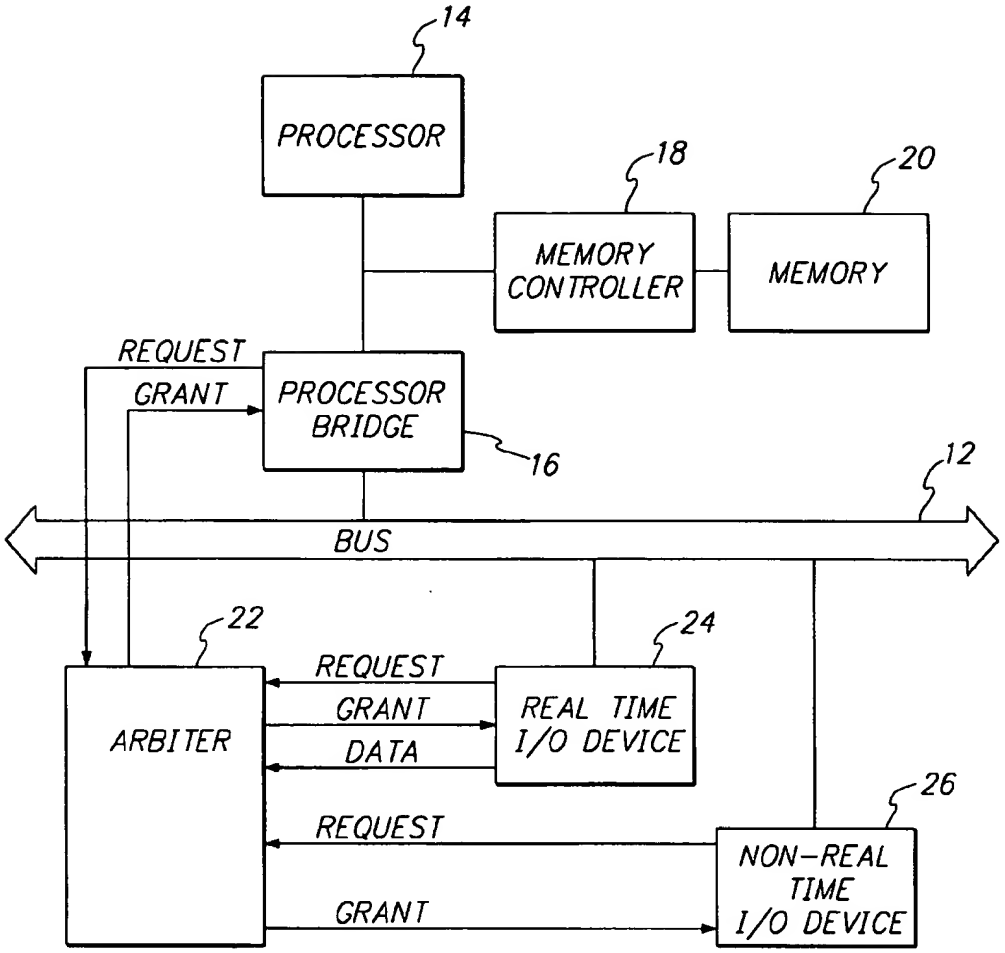


FIG. 1

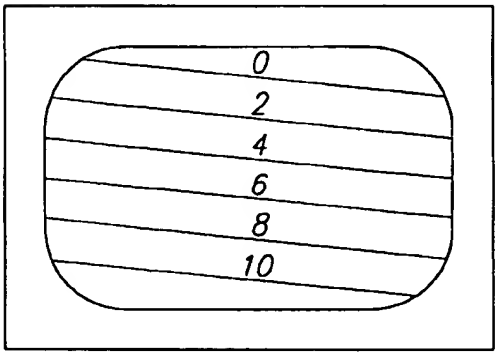


FIG. 2(a)

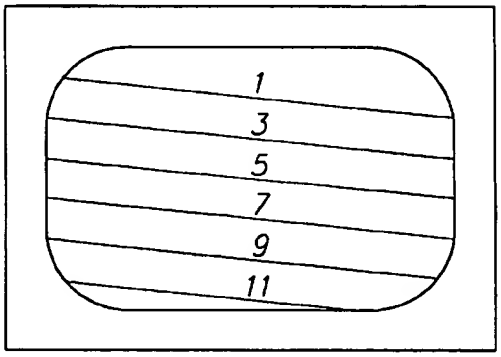


FIG. 2(b)

FIG. 3

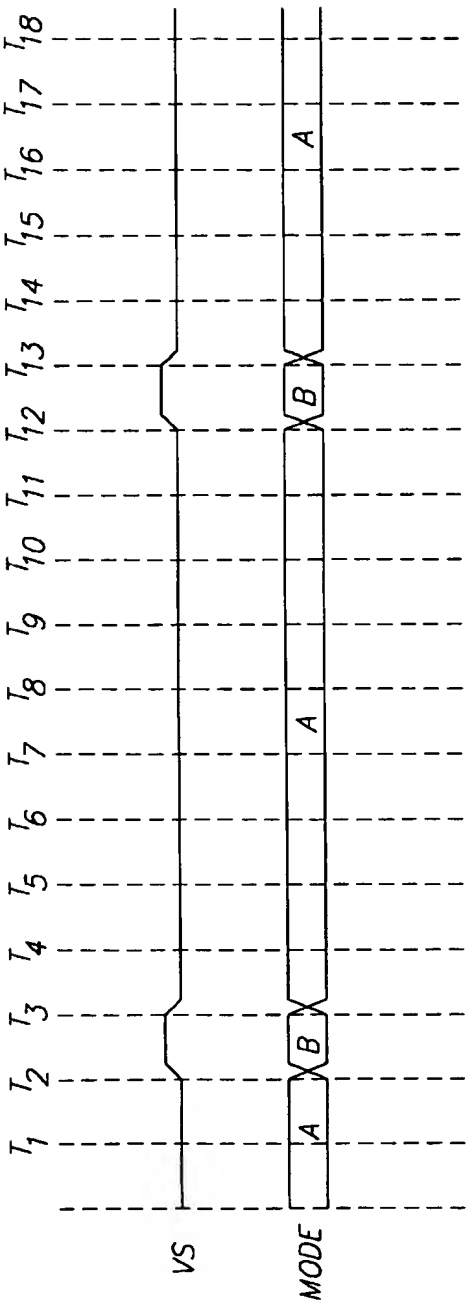
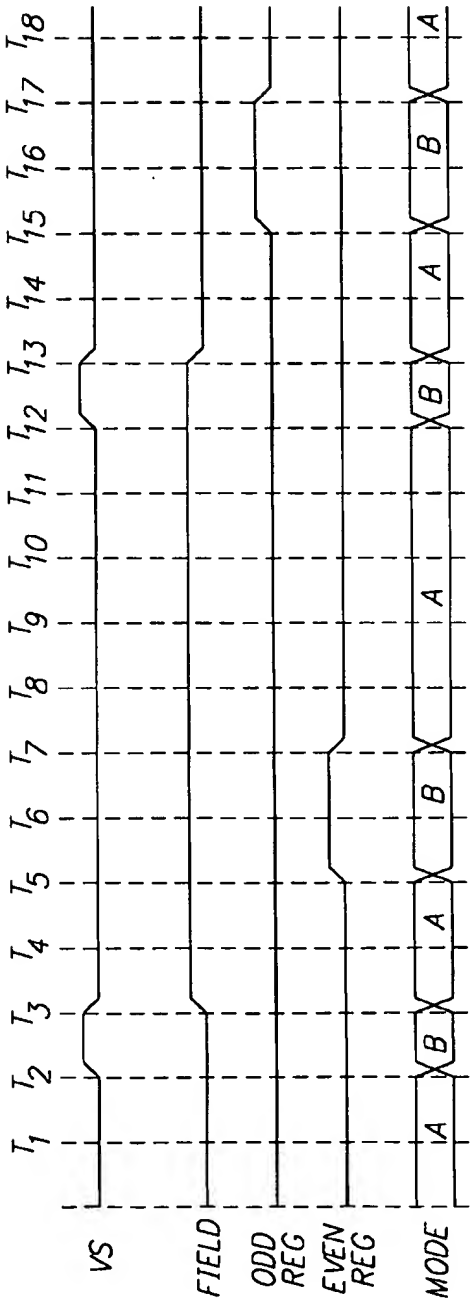

T1 T2 T3 T4 T5 T6 T7 T8 T9 T10 T11 T12 T13 T14 T15 T16 T17 T18
VS
MODE
A
B

FIG. 5


T1 T2 T3 T4 T5 T6 T7 T8 T9 T10 T11 T12 T13 T14 T15 T16 T17 T18
VS
FIELD
ODD REG
EVEN REG
MODE
A
B

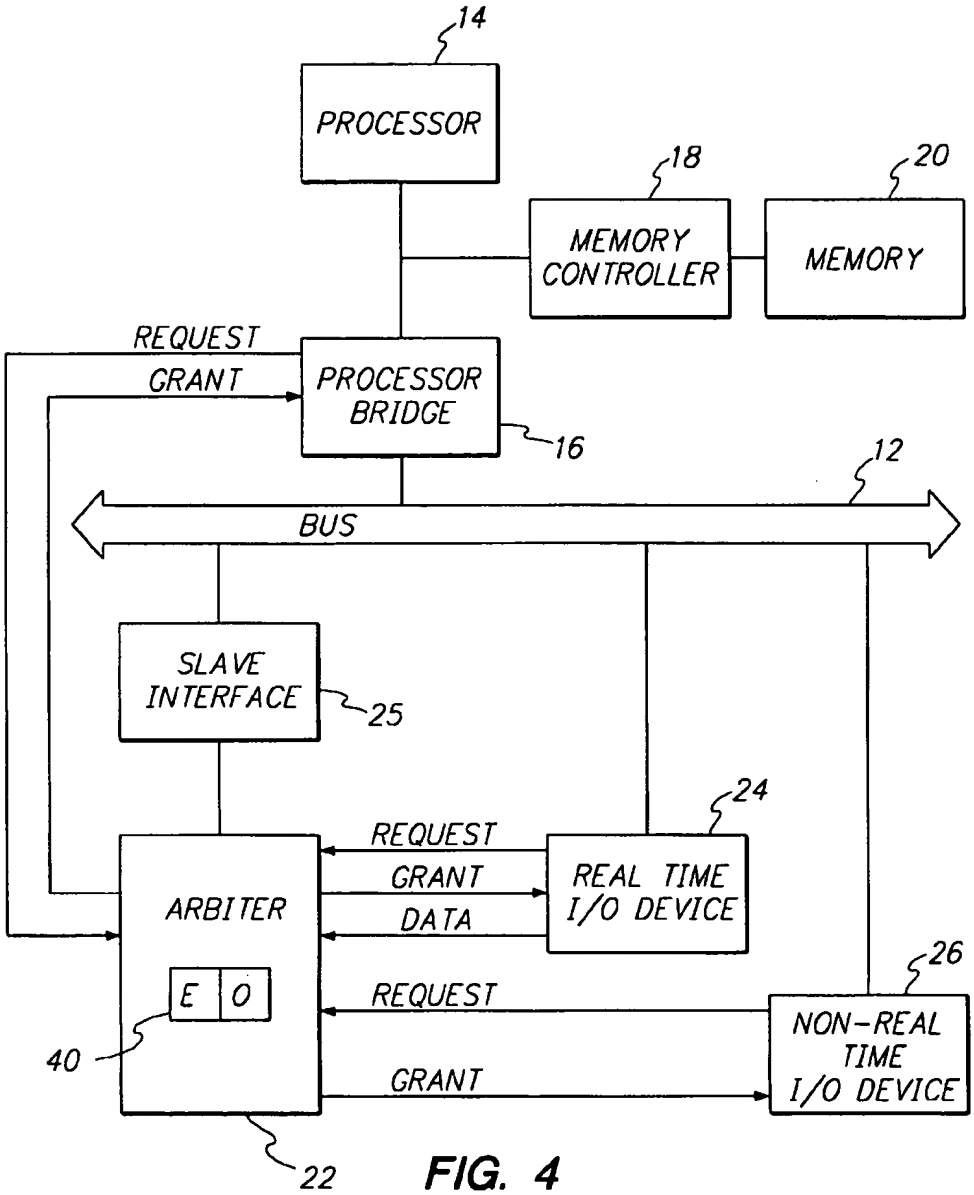

14
PROCESSOR
18
MEMORY CONTROLLER
20
MEMORY
REQUEST
GRANT
PROCESSOR BRIDGE
16
12
BUS
SLAVE INTERFACE
25
24
REQUEST
GRANT
DATA
REAL TIME I/O DEVICE
ARBITER
E
O
40
26
REQUEST
NON-REAL TIME I/O DEVICE
GRANT
22

FIG. 4

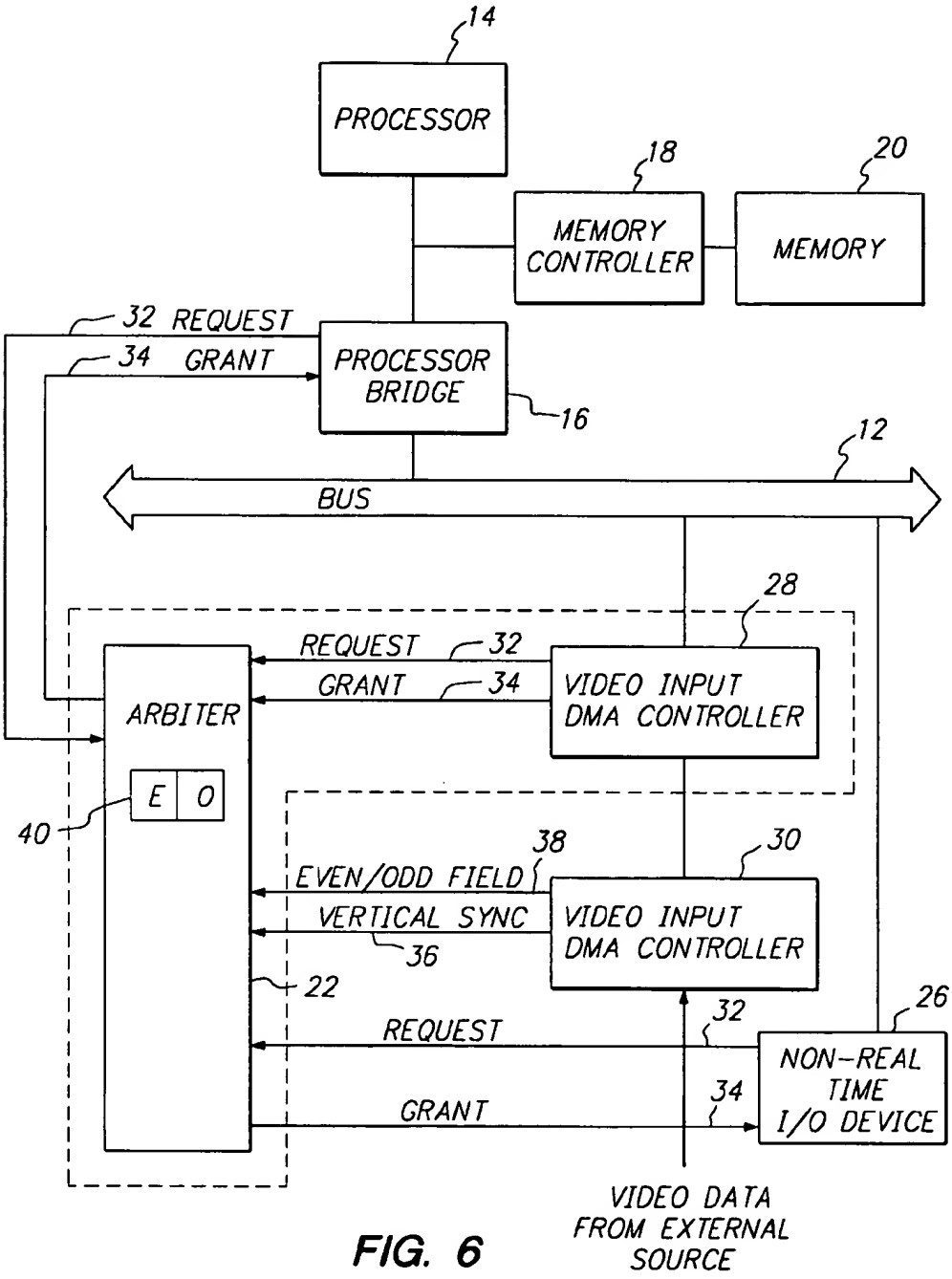

14
PROCESSOR
18
MEMORY CONTROLLER
20
MEMORY
32 REQUEST
34 GRANT
PROCESSOR BRIDGE
16
12
BUS
28
REQUEST 32
GRANT 34
ARBITER
VIDEO INPUT DMA CONTROLLER
E O
40
38
30
EVEN/ODD FIELD
VERTICAL SYNC
36
VIDEO INPUT DMA CONTROLLER
22
26
32
REQUEST
NON-REAL TIME I/O DEVICE
34
GRANT
VIDEO DATA FROM EXTERNAL SOURCE

FIG. 6

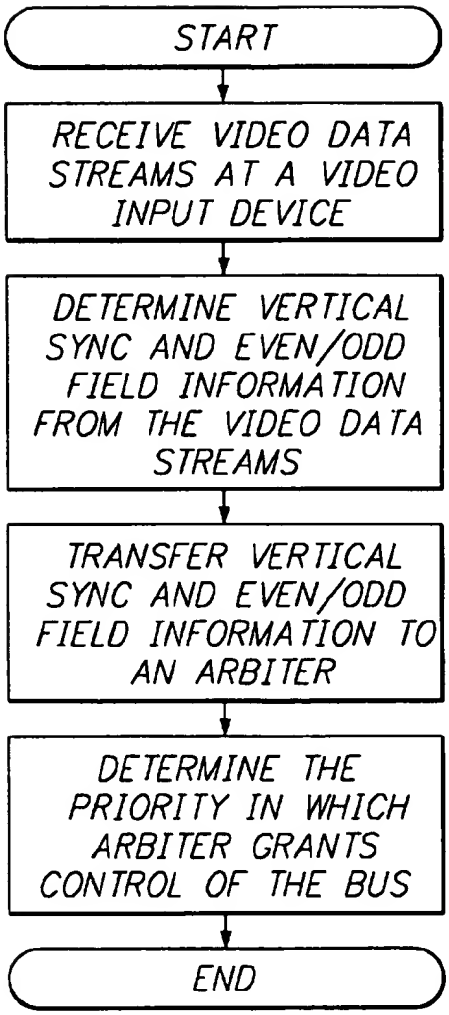

START
RECEIVE VIDEO DATA STREAMS AT A VIDEO INPUT DEVICE
DETERMINE VERTICAL SYNC AND EVEN/ODD FIELD INFORMATION FROM THE VIDEO DATA STREAMS
TRANSFER VERTICAL SYNC AND EVEN/ODD FIELD INFORMATION TO AN ARBITER
DETERMINE THE PRIORITY IN WHICH ARBITER GRANTS CONTROL OF THE BUS
END

FIG. 7

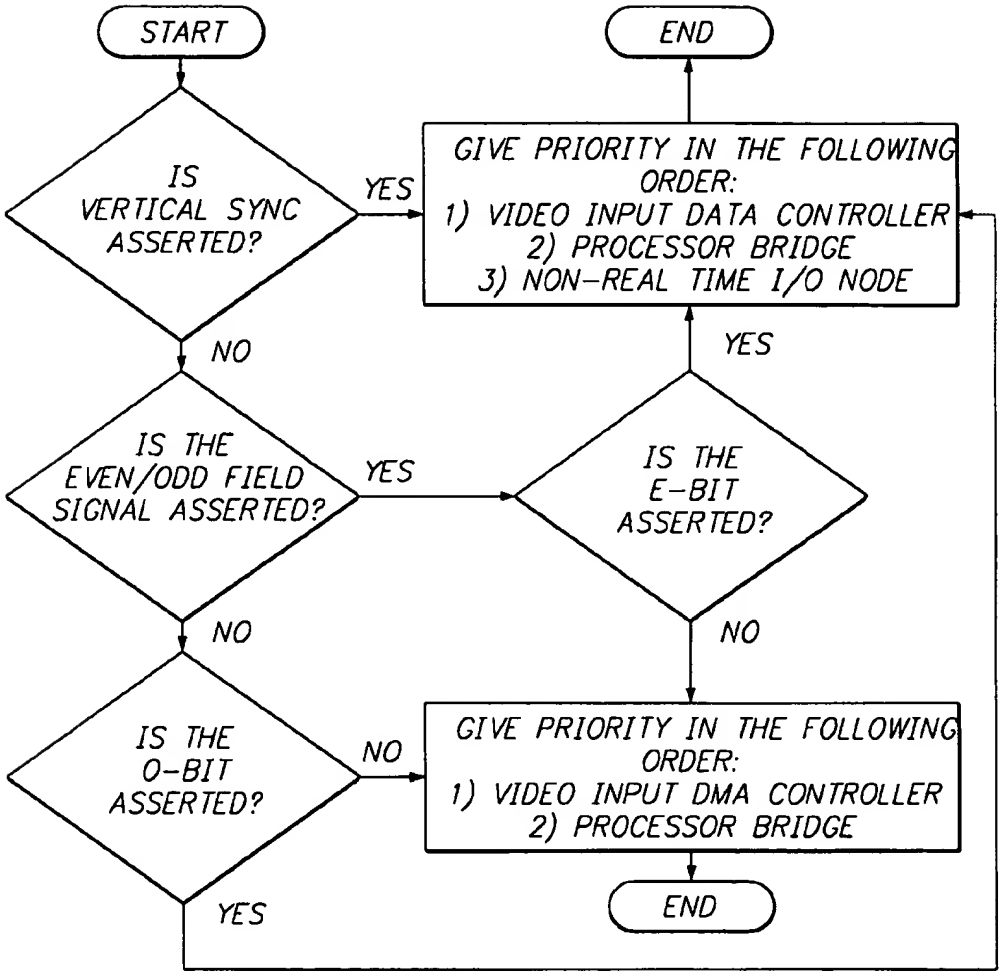

START
END
IS VERTICAL SYNC ASSERTED?
YES
GIVE PRIORITY IN THE FOLLOWING ORDER:
1) VIDEO INPUT DATA CONTROLLER
2) PROCESSOR BRIDGE
3) NON-REAL TIME I/O NODE
NO
YES
IS THE EVEN/ODD FIELD SIGNAL ASSERTED?
YES
IS THE E-BIT ASSERTED?
NO
NO
IS THE O-BIT ASSERTED?
NO
GIVE PRIORITY IN THE FOLLOWING ORDER:
1) VIDEO INPUT DMA CONTROLLER
2) PROCESSOR BRIDGE
YES
END

FIG. 8

FIG. 9

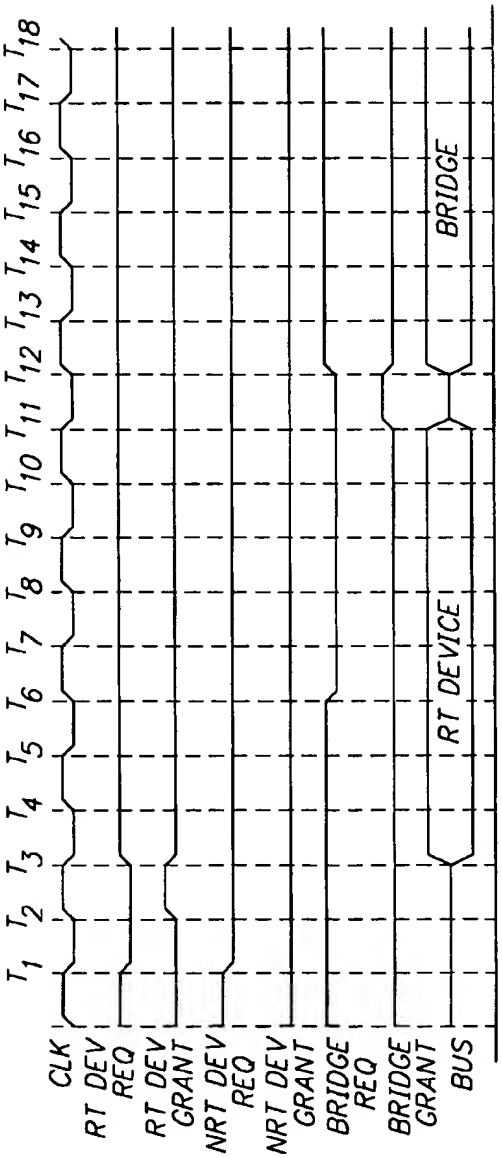


FIG. 10

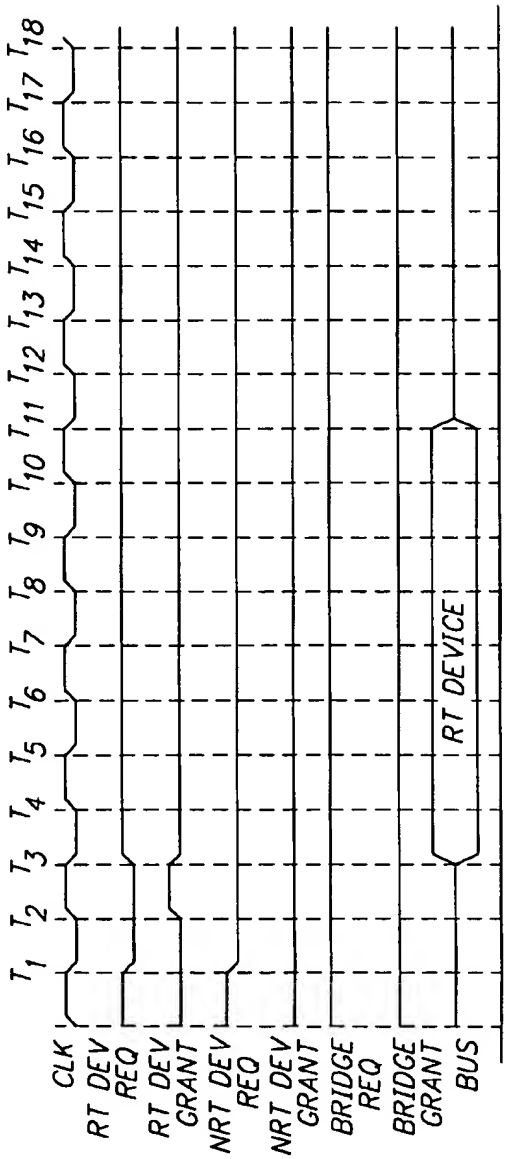


FIG. 11

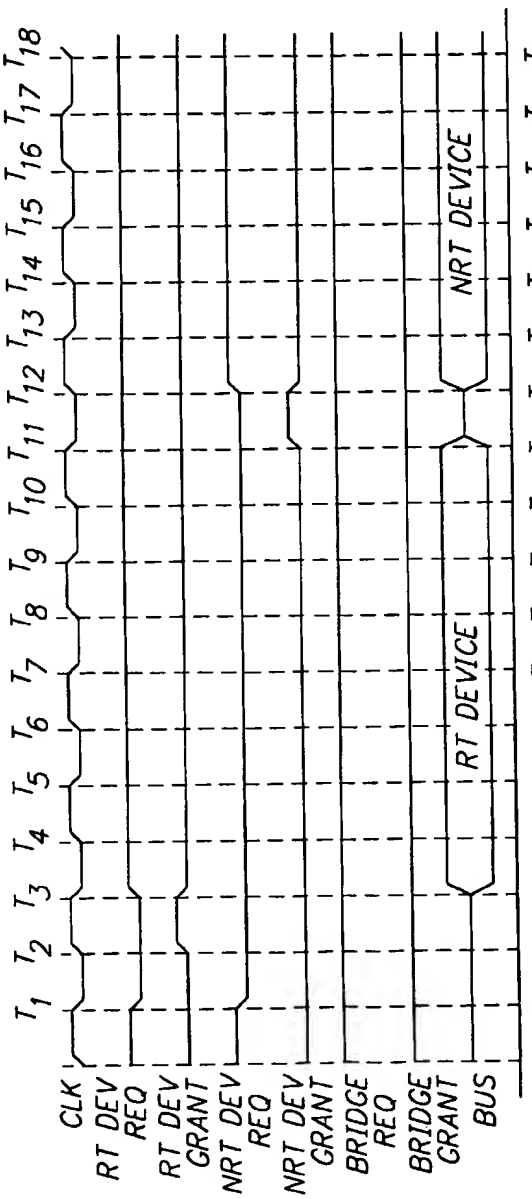
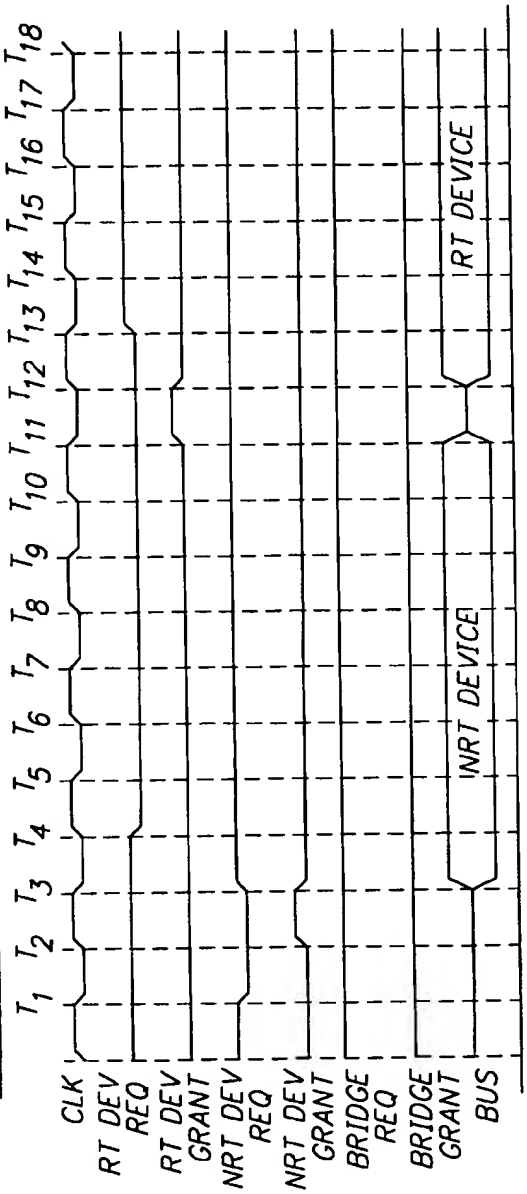


FIG. 12



APPARATUS FOR AND METHOD OF ARBITRATING BUS CONFLICTS

This application is a continuation-in-part application of application Ser. No. 08/436,968, filed May 8, 1995, now abandoned

FIELD OF THE INVENTION

The present invention relates to bus arbitration in computer systems, and more particularly, to a bus arbitration system in which an arbiter is responsive to a signal indicative of the status of information for transfer by a device on the bus to set bus access priority, and to a bus arbitration system in which a bus arbiter can be programmed or configured by the system processor to change its mode of prioritization.

BACKGROUND OF THE DISCLOSURE

In many computer systems, components are connected by a shared bus. In such an architecture, only one unit at a time can exert control over the shared bus. Contention among the units which request use of the bus at the same time must be resolved in some manner, i.e., arbitrated.

The most basic arbitration scheme is the so-called "first in first out" arbitration scheme. In this scheme, requests to use the bus are granted strictly in the order in which they arise. This scheme has the advantage of simplicity, but the disadvantage that the relative importance of a device is not taken into account in determining when to grant its request for access to the bus. Thus, devices urgently requiring access to the bus may have to wait while other lower priority devices complete their use of the bus.

A commonly used prior art method for bus arbitration overcomes the above disadvantage by using a central arbiter communicating with all of the devices requesting use of the bus. The central arbiter determines which device has the highest bus use priority taking into account relative importance of the device and how long the device has been waiting to use the bus.

Another known arbitration system can be referred to as "daisy chaining". In this method, a signal indicating that the bus is available for use is transmitted from device to device in priority order. Any device requiring access to the bus captures the signal during its turn and uses the bus. Another prior art scheme offers a rotating priority method wherein bus access is offered to each device in a strict rotation.

The demands for efficient arbitration of bus contentions are especially pronounced in a computer system which processes real time data streams, such as video data streams, and audio data streams, animation data streams. It is crucial for real time data streams to be transferred without interruption since an interruption may be visually or audibly noticeable to an observer. Furthermore, if an arbiter is unable to provide a relatively rapid grant to a device transferring real time data streams, large buffers within the device may be required to store the real time data before it can be transferred. For example, if the arbiter were not assuredly able to provide a relatively swift grant to a video input DMA controller, the video input DMA controller would need large data FIFO buffers to detain high bandwidth video input until it can be transferred via the bus. Large FIFO buffers add significant size and thus cost to the design of a video input controller.

As a result, processing real time data streams requires special control of the system bus beyond what is necessary for non-real time data streams. In some prior art systems, this need is addressed by permitting a real time data stream controller such as a video input DMA controller to retain exclusive use of the bus once granted control until the entire video data stream has been transferred.

SUMMARY OF THE INVENTION

It is an object of the present invention to overcome the deficiencies of the prior art by providing a method of and apparatus for prioritizing the allocation of a bus to a plurality of devices during a real-time data transfer. The invention takes advantage of intervals during an overall data transfer operation when in fact no active data is being transferred. In a system according to the invention, an arbiter is given information about such intervals and can permit other devices to use the bus during the intervals. In another aspect of the invention, the arbiter receives arbitration information from the system which the arbiter uses to determine a prioritization mode.

In one aspect, the present invention is an arbitration system in which a bus arbiter determines a priority of granting control of the bus based at least in part on a data status signal from one of the devices connected to the bus. In one embodiment, the device transfers real time data in the form of a video signal, and the data status signal indicates whether the concurrent video signal is a synchronization signal. The arbiter can additionally or alternatively determine priority in accordance with programming data supplied by the system processor. The programming data may indicate a concurrent system requirement such as for odd field or even field data. The data status signal may additionally or alternatively indicate which of even field data and odd field data constitutes the concurrent video signal.

In yet another aspect, the invention is a method of arbitrating bus conflicts between a first device and a second device in a computer system, the method comprising the step of determining a priority of granting control of the bus to one of the first device and the second device based at least in part on a data status signal from the first device. Priority may additionally or alternatively be determined in accordance with programming data supplied by the system processor.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features and advantages of the invention will be readily apparent to one of ordinary skill in the art from the following written description, used in conjunction with the drawings, in which:

FIG. 1 is a functional block diagram of a system incorporating one embodiment of the present invention;

FIGS. 2(*a*) and 2(*b*) illustrate a video scanning pattern;

FIG. 3 is a timing diagram showing how a prioritization mode might be set in accordance with the presence or absence of a vertical sync signal;

FIG. 4 is a functional block diagram of a system incorporating a second embodiment of the present invention;

FIG. 5 is a timing diagram showing how a prioritization mode might be set in response to data status signals;

FIG. 6 is a functional block diagram of a system incorporating a third embodiment of the present invention;

FIG. 7 is a flowchart illustrating instruction flow for a system according to one embodiment of the present invention;

FIG. 8 is a flowchart illustrating instruction flow for a programmable arbiter according to one embodiment of the present invention; and

FIG. 9–12 are timing diagrams showing various arbitration scenarios.

DETAILED DESCRIPTION

FIG. 1 is a functional block diagram of a computer system incorporating one embodiment of the present invention. The system in FIG. 1 includes a system bus 12. In the presently preferred embodiment, this bus is a Peripheral Component Interconnect (PCI) local bus. Details concerning PCI local buses are provided in "PCI Local Bus Specification," Review Draft Revision 2.1, published Oct. 21, 1994, by PCI Special Interest Group. It will be readily appreciated by one skilled in the art, however, that the present invention is not limited to use in systems having a PCI local bus, but rather can be used with any high performance bus.

As illustrated in FIG. 1, the system bus 12 has three nodes. One node is I/O device 24 which in this example is a node over which real time data is transferred. Another node is I/O device 26 which is a node which will transfer exclusively non-real time data. A third node is the processor bridge 16, through which system components such as a processor 14, a memory controller 18, and a memory 20 access the bus 12. It should be noted, however, that it is possible to connect a processor directly to the bus, in which case references to processor bridge as used herein should be understood as including the case of connection to the processor directly.

These devices are all bus master devices, i.e., devices which request control of the system bus to write or read data to or from another (slave) device via the system bus 12. It will be appreciated by one skilled in the art that the present invention is not limited to a system which contains only three nodes, but instead has application to systems having any one of a number of alternate configurations.

The processor bridge 16 allows the processor 14 to access the devices connected to the PCI bus. One example of this type of access is when the processor 14 performs read or write operations to registers that are contained in either the real time I/O device 24 or the non-real time I/O device 26. The processor bridge 16 also allows either of the I/O devices 24 and 26 to access the system memory 20 via the memory controller 18 in a known manner.

The system of FIG. 1 also includes an arbiter 22. The arbiter 22 is connected to the real time I/O device 24, the non-real time I/O device 26, and the processor bridge 16 by respective pairs of lines, each pair containing a "REQUEST" line and a "GRANT" line. These are the lines over which the I/O devices and bridge send requests to use the system bus 12 to the arbiter 22, and over which the arbiter 22 sends a signal granting the request if the arbitration scheme used by the arbiter 22 permits it.

In the embodiment shown, the arbiter 22 also receives at least one additional signal ("DATA") from the real time I/O device 24. This signal is indicative of a status of the data stream being transferred through the real time node 24. For example, if the real time node 24 is transferring video data, the DATA signal may be a signal indicating that the current portion of the video data is a vertical sync signal, or a horizontal sync signal, or whether an even or odd field of a video frame is being transferred. Although the DATA line shown here is a single line, it will be understood that the DATA signal may include additional lines.

It is one feature of the present invention that the arbiter 22 can use this information in its prioritization of access requests. In other words, the arbiter 22 is able to employ one prioritization scheme during one portion of the real time data stream, and another prioritization scheme during another portion of the real time data stream.

For example, suppose the data signal indicates the existence of a vertical sync pulse. The derivation of a digital video sync signal from a video signal is conventional. During a sync pulse, horizontal or vertical, real time data demands on system resources are reduced since no active video data is being transferred. The reason for this may be understood by referring to FIGS. 2(*a*) and 2(*b*). Video scanning starts at the upper left hand corner of an image and proceeds to the right with a slightly downward motion as illustrated. When the scan reaches the right side of the image it rapidly returns or retraces to a position below its starting point of the previous line and again proceeds to the right in a slightly downward motion. The time it takes the scan beam to retrace back to the left side is called horizontal sync (HS) time.

A succession of active scans and rapid retraces continues until a point near the bottom of the picture is reached, at which time the beam rapidly moves upward to a mid-point at the top of the picture. When the scan reaches the bottom of a picture, it must return to the top of the picture before the next scanning line can begin. The time it takes for the scan beam to go from the bottom of the picture to the top of the picture is called vertical sync (VS) time.

The downward scan and retrace motions are then repeated, with the second set of scan lines falling between the first set of scan lines. The first set of scan lines is called an even scan line field and the second set of scan lines is called an odd scan line field.

As a result, a video data stream is comprised of a plurality of alternating even scan lines and horizontal sync periods followed by a vertical sync period which in turn is followed by a plurality of alternating odd field scan lines and horizontal sync periods followed by a last vertical sync period. During the vertical sync and horizontal sync periods, the scan line merely returns to a new starting position so no active video data is generated during these periods.

It is therefore possible to use a prioritization scheme during such periods in which non-real-time data can have a higher priority. This is depicted in FIG. 3, in which the arbitration scheme or mode switches between a mode A during active video (non-sync) time and a mode B where the vertical sync pulse is present (no active video).

In mode A the non-real-time data is given less or even no priority. In mode B the non-real-time data is given a higher priority than it has in mode A. In the presently preferred embodiment, in mode A, access requests are prioritized with requests from the real time node being given the highest priority, requests from the processor bridge being given the next priority, and requests from the non-real-time node being given no priority at all so that the non-real-time node's access to the system bus is effectively blocked. In mode B, access requests are prioritized with requests from the real time node being given the highest priority, requests from the processor bridge being given the next priority, and requests from the non-real-time node being the next priority. Thus, the difference between mode A and mode B is that in mode A the non-real-time device has no opportunity for access to the system bus, while in mode B the non-real-time device is given an opportunity for access to the bus.

In the above scheme, the real time I/O device is always given first priority. In the example of a video I/O device, for example, this is because video input controller may have to use the bus during vertical sync time as well as during active video time for tasks such as reporting the status of the previous video field and setting up the next video field. These tasks nevertheless must be completed before the

active data for the next video field starts being transferred. Thus, "real time" demands cannot be assumed to be completely absent during the sync portion of the signal, but they can be assumed to be reduced and less critical.

It should also be noted that the DATA signal is derived from the highest priority device. Since servicing this device is paramount, it is the status of data to be transferred by this device which will in general determine whether other devices should be given an opportunity to access the bus.

In general, once a device has won an arbitration, it is permitted to maintain its control of the system bus until it has completed the data transfer operation for which it requested access. Thus, it is possible in mode B that a non-real-time I/O device has control of the system bus when the real time I/O device requests access. In such circumstances, the real time I/O device would have to wait. The possible adverse effects of this conflict can be mitigated, however, by imposing a requirement that the processor bridge accept only data transfers having a length less than or equal to a certain set number of data phases. This limit may be, say, eight data phases, which would be one cache line on the processor side (256 K cache with a 32 byte size). At this limit, the processor bridge disconnects, and the non-real-time node must re-arbitrate for the bus if it needs to, thus giving the real time device a chance to assert its higher priority. While theoretically eight data phases can take infinitely long to complete if wait states are inserted, as a practical matter there are recommended guidelines for data latency and most if not all devices will not arbitrate for the bus until they can source or sink all of the data for a given transfer. Similar considerations apply to non-real-time data transfers overlapping the transition from mode B to mode A.

As suggested above, there are other intervals where no active data is being transferred, such as the horizontal sync period in a video signal, where non-real-time devices might be provided an opportunity to arbitrate for access to the bus. There are also intervals where active data is being transferred but is not needed by the system, and so where non-real-time devices might be provided an opportunity to arbitrate for access to the bus. For example, for video, there are operations (such as display of a reduced-size image) which may require only one of the odd or even data fields from a video data stream. The other data field does not have to be transferred in such circumstances. As a result, there are intervals of "off-field transfer" when the real time I/O device's need for the bus is reduced.

Since the system's demand for one, both, or neither of the data fields is conditional, it is advantageous to provide an arbiter which can alter its mode of operation in response to information about system demand. In effect, this system demand information programs or configures the arbiter to alter the arbiter's response to the data status signal. In particular, and using the example from above, the arbiter can revert to mode B operation when off-field data is being transferred, thus permitting the non-real-time I/O device access to the bus which it otherwise would not have. Such a system is shown in FIG. 4. In FIG. 4, the arbiter 22 is connected to the system bus 12 through a slave interface 25, thus permitting the processor to write programming information to storage in the form of a control register 40 within the arbiter 22. Note that the arbiter 22 of FIG. 1 is not necessarily connected to the bus 12 and can be anywhere in the system.

Continuing with the example of video as a real time data stream, in a presently preferred embodiment, the programming information indicative of system requirements is a signal which sets single bit values ("E" and "O") to be stored in the control register 40. The value of "E" is indicative of system requirements for the even video field while the value of "O" is indicative of system requirements for the odd video field. The values assigned to each condition are arbitrary, but one scheme is shown in the following table:

E	O	REQUIREMENT
0	0	BOTH
0	1	EVEN ONLY
1	0	ODD ONLY
1	1	NEITHER

Once the arbiter is programmed with this information, it can switch to, for example, the B mode when an off-field is being transferred. This is shown in FIG. 5, which shows a case where the system requirements change during transmission of a field. This case is presented purely for the sake of illustrating the principles of operation of the invention since it will be readily understood by one of ordinary skill in the art that system configuration will normally be relatively static in use. That is, normally the contents of the register 40 will remain set for time periods on the order of minutes or even longer, and one would in practice very rarely if ever observe a case where those contents would change during a vertical sync.

In FIG. 5, a high "FIELD" signal means that the even field is currently being transferred, and a low "FIELD" signal means an odd field is being transferred. As can be seen, the arbiter 22 determines a mode of operation based on the status of the data stream in accordance with the programming data stored in its control registers.

Another example of a system according to the present invention is illustrated in FIG. 6. The system of FIG. 6 again uses video data as an example of real time data, but it will still be understood by one having ordinary skill in the art that other types of real time data can be handled in a similar fashion.

FIG. 6 illustrates several bus master devices including a processor bridge 16, a video input DMA controller 28 and a non-real time I/O device 26 which are connected to the bus 12. Each bus master device is also connected to a programmable arbiter 22 by a request line 32 and a grant line 34. Each bus master device can request control of the bus by asserting the request line. In tarn, the arbiter 22 can grant control of the bus 12 by asserting the grant line of a requesting bus master device. The arbiter 22, which preferably may be made up of a suitable arrangement of gate and logic circuits in a manner which will be apparent to one having ordinary skill in the art, contains a control register 40 the contents of which control the arbiter operation.

In the implementation illustrated in FIG. 6, the arbiter is shown as being part of the video input DMA controller device 28. This allows the processor 14 to utilize the PCI slave portion of the video input DMA controller 28 to gain access to the control register 40 present in the arbiter 22. A DMA controller is more completely described in U.S. patent application Ser. Nos. 08/340,248 and 08/340,249, both of which are incorporated herein by reference.

In the embodiment illustrated, the arbiter assigns the video input DMA controller a higher priority than the other bus master devices, since the video input DMA controller handles real-time video signals. However, as noted above, even during a video data stream transfer there are periods of time in which there is no active video data to be transferred.

According to the present invention, the arbiter **22** can be reprogrammed to change the arbitration scheme during the transfer of a real-time data stream.

Specifically, the video data streams are inputted into a video input device **30** from an external device. As shown in FIG. **7**, the video input device **30** detects a plurality of video related signals from the video data stream, i.e., vertical sync, horizontal sync, and even/odd field, in a known manner. The video input device **30** then transfers the video related signals to the arbiter **22**. In one embodiment of the present invention, the video input device **30** transfers the vertical sync and even/odd field information to the arbiter **22**. In addition, the video input device **30** passes the video data streams to the video input DMA controller **28**.

The arbiter **22** uses the two video timing signals and the contents of the control register **40** to determine the priority of the bus master devices during the transfer of video data streams. The arbiter control register **40** is a two bit register with one bit, an E-bit, controlling the response of the arbiter **22** during an "even" field portion of the video data stream and the other bit, an O-bit, controlling the response of the arbiter **22** during the "odd" field portion of the video data stream. The data stored in the arbiter control register **40** is supplied by the system processor, and indicate to the arbiter **22** whether the overall system needs either, both, or neither of the even or odd fields of the current frame.

The arbiter **22** thus has data from the video input device **30** indicating when the vertical sync occurs and if an even or odd field is being input. The arbiter **22** also has data indicating whether one or both of the even and odd field is superfluous. The arbiter **22** can use this data to identify intervals during real time video transfer when the video input DMA controller has a reduced need to use the bus **12** and to permit other devices to use the bus **12** during such intervals.

A flowchart for arbiter operation is shown in FIG. **8**. If the vertical sync signal is being asserted, i.e. the video data stream has reached a vertical sync period, the arbiter gives the highest priority for access to the bus to the video input DMA controller **28** followed by the processor bridge **16** and then the non-real time device **26**.

If the vertical sync signal is deasserted and the even/odd field signal is asserted, i.e., the video signal is in an even field period, the arbiter checks the status of the E-bit. If the E-bit is set to zero, meaning that the system intends to use the even field of the video signal, the arbiter **22** gives the highest priority to the video input DMA controller followed by the processor bridge **16** and does not permit access by the non-real time node. If the E-bit is set to **1**, meaning the system does not intend to use the even field portion of the video data, the arbiter gives the highest priority to the video input DMA controller followed by the processor bridge and then the non-real time I/O node.

If the vertical sync signal is deasserted and the even/odd field signal is deasserted, i.e., the video signal is in an odd field period, the arbiter checks the status of the O-bit. If the O-bit is set to zero, meaning that the system intends to use the odd field, the arbiter gives the highest priority to the video input DMA controller followed by the processor through the bridge **16**. In addition, the arbiter does not permit access to the non-real time node. As above, the premise is that video data is critical, so that only an interruption originating from the processor **14** will be tolerated. Interruptions originating from a presumptively less critical non-real-time device are not permitted. If the O-bit is set to 1, the arbiter gives the highest priority to the video input DMA controller followed by the processor bridge and then the non-real time I/O node.

FIGS. **9**, **10**, **11**, and **12** are timing diagrams showing operation of the arbitration system as described above. FIG. **9** shows operation of the system in mode A or mode B. The upper trace is the system clock. In the figure, the assumption has been made that data phases correspond to clock pulses, which is true in a well-behaved system. Times T1–T18 are designated in the top row. It is important to note, however, that these times do not correspond to the times in the preceding figures relating to video signals. As is well-known, the duration of the clock pulses is exceedingly small compared to the time of transmission of a video field.

FIG. **9** shows the example of a real-time device and a non-real-time device simultaneously requesting access to the bus. In mode A, the non-real-time device request would simply be ignored; in mode B, the real-time device would win an arbitration. Thus, in either case, the system would respond by providing a real-time device grant signal to the real-time device. This is shown at times T2 to T3. The non-real-time device continues to assert its request, however. In the meantime, however, the bridge requests access to the bus at time T6. Thus, since the real-time device ceases control of the bus after eight data phases at time T1, and does not assert a new request for access, the bridge request is granted, and the bridge assumes control of the bus. Again, this would occur in mode A because the continuing request for the bus from the non-real-time device would be ignored, or in mode B because the bridge would win an arbitration with the non-real-time device for access to the bus.

FIG. **10** shows operation in mode A where the real-time device and the non-real-time device simultaneously request access to the bus at a time T1. Again, the real-time device uses the bus but does not reassert a request for access to the bus before it is finished. Although the non-real-time device has a request for access to the bus pending, however, it is ignored since in mode A the arbiter never permits the non-real-time device access to the bus.

FIG. **11** shows operation in mode B. Operation is essentially the same as in mode A up to time T11. At time T11, however, the arbiter does grant the non-real-time device access to the bus by returning a grant signal. The non-real-time device then asserts control of the bus for up to eight data phases.

FIG. **12** shows operation again in mode B where the non-real-time device requests access to the bus at a time when no other requests are pending. The arbiter therefore awards control of the bus to the non-real-time device. While the non-real-time device has control of the bus, however, the real-time device asserts a request for access at time T4. In this circumstance, however, the real-time device must wait until the non-real-time device has finished using the bus, at which time the real-time device can then be given control of the bus.

These are exemplary. Other situations, of course, arise. For example, it is possible that in mode A, the processor bridge has control of the bus when the real-time device requests access. Again, in such a circumstance, the real-time device must wait until the processor bridge finishes using the bus, at which time the real-time device can be given control of the bus.

It will be appreciated by those of ordinary skill in the art that the present invention can be embodied in other specific forms without departing from the spirit or essential character thereof. The presently disclosed embodiments are therefore considered in all respects to be illustrated but not restrictive.

The scope of the invention is indicated by the appended claims rather than the foregoing description, and all changes which come within the meaning and range of equivalence thereof or intended to be embraced therein.

I claim:

1. A computer system comprising:

a bus:

first device coupled to said bus and transferring video data;

a second device coupled to said bus and transferring non-real time data;

a bus arbiter coupled to said first device and said second device, said bus arbiter determining a priority of granting control of said bus to one of said first device and said second device based at least in part on a data status signal from said first device.

2. A computer system as claimed in claim 1 wherein said data status signal indicates whether the concurrent video signal is a vertical synchronization signal.

3. A computer system as claimed in claim 1 wherein said data status signal indicates whether the concurrent video signal is a horizontal synchronization signal.

4. A computer system as claimed in claim 1 wherein said bus arbiter operates in one of a first mode when said data status signal indicates that the concurrent video signal is not a synchronization signal, wherein bus access requests from said second device are not granted, and a second mode when said data status signal indicates that the concurrent video signal is a synchronization signal wherein bus access requests from said second device may be granted.

5. A computer system as claimed in claim 1 wherein said computer system further includes a processor bridge coupled to said bus and wherein said bus arbiter operates in a first mode when said data status signal indicates that the concurrent video signal is not a synchronization signal in which bus access requests from said first device are given priority over bus access requests from said processor bridge and bus access requests from said second device are not granted, and in a second mode when said data status signal indicates that the concurrent video signal is not a synchronization signal in which bus access requests from said first device are given priority over bus access requests from said processor bridge, which in turn are given priority over bus access requests from said second device.

6. A computer system as claimed in claim 1 further comprising a processor coupled to said bus, wherein said bus arbiter is coupled to said bus and determines a priority of granting control of said bus to one of said first device and said second device additionally in accordance with programming data supplied by said processor.

7. A computer system as claimed in claim 6, wherein said first device transfers real time data.

8. A computer system as claimed in claim 7, wherein said second device transfers non-real time data.

9. A computer system as claimed in claim 8, wherein said real time data is a video signal, and wherein said data status signal indicates which of even field data and odd field data constitutes the concurrent video signal.

10. A computer system as claimed in claim 9 wherein the arbiter includes a memory for storing said programming data.

11. A computer system as claimed in claim 9 wherein said programming data indicates a concurrent system requirement for odd field data.

12. A computer system as claimed in claim 9 wherein said programming data indicates a concurrent system requirement for even field data.

13. A computer system as claimed in claim 12 wherein said programming data indicates a concurrent system requirement for odd field data.

14. A computer system as claimed in claim 13 wherein said bus arbiter operates in one of a first mode when said data status signal indicates that the concurrent video signal is not a synchronization signal and said data status signal indicates that a field being transferred is one for which the programming data indicates a concurrent system requirement, wherein bus access requests from said second device are not granted, and a second mode when said data status signal indicates that the concurrent video signal is a synchronization signal or said data status signal indicates that a field being transferred is one for which the programming data indicates no concurrent system requirement wherein bus access requests from said second device may be granted.

15. A method of arbitrating bus conflicts between a first device and a second device in a computer system, said method comprising the step of determining a priority of granting control of said bus to one of said first device which transfers real time data in the form of a video signal and said second device based at least in part on a data status signal from said first device indicating that the concurrent video signal is a synchronization signal.

16. A method as claimed in claim 15 wherein bus access requests from said second device are not granted when said data status signal indicates that the concurrent video signal is not a synchronization signal and wherein bus access requests from said second device may be granted when said data status signal indicates that the concurrent video signal is a synchronization signal.

17. A method as claimed in claim 15 wherein said computer system further includes a processor bridge coupled to said bus and wherein bus access requests from said first device are given priority over bus access requests from said processor bridge and bus access requests from said second device are not granted when said data status signal indicates that the concurrent video signal is not a synchronization signal and wherein bus access requests from said first device are given priority over bus access requests from said processor bridge, which in turn are given priority over bus access requests from said second device when said data status signal indicates that the concurrent video signal is not a synchronization signal.

18. A method as claimed in claim 15 wherein said data status signal indicates that the concurrent video signal is a vertical synchronization signal.

19. A method as claimed in claim 15 wherein said data status signal indicates that the concurrent video signal is a horizontal synchronization signal.

20. A method as claimed in claim 19 wherein said computer system includes a processor, and wherein determining step further comprises determining a priority of granting control of said bus to one of said first device and said second device additionally in accordance with programming data supplied by said processor.

21. A method as claimed in claim 20 wherein said real time data is a video signal, and wherein said data status signal indicates which of even field data and odd field data constitutes the concurrent video signal.

22. A method as claimed in claim 21 further comprising the step before said determining step of storing said programming data.

23. A method as claimed in claim 21 wherein said programming data indicates a concurrent system requirement for odd field data.

24. A method claimed in claim 21 wherein said programming data indicates a concurrent system requirement for even field data.

25. A method as claimed in claim 24 wherein said programming data indicates a concurrent system requirement for odd field data.

26. A method as claimed in claim 25 wherein bus access requests from said second device are not granted when said data status signal indicates that the concurrent video signal is not a synchronization signal and said data status signal indicates that a field being transferred is one for which the programming data indicates a concurrent system requirement, and wherein bus access requests from said second device may be granted when said data status signal indicates that the concurrent video signal is a synchronization signal or said data status signal indicates that a field being transferred is one for which the programming data indicates no concurrent system requirement.

* * * * *